# COMPUTE

the Club Of Microprocessor Programmers, Users, and Technical Experts

Georgia Marszalek, Editor • David Graves, Editor

**Sponsored by National Semiconductor Corp., Santa Clara, Ca. 95051** **Vol. 2, No. 7, July, 1976**


# NATIONAL ANNOUNCES "8080A" MICROPROCESSOR

The well-known model 8080A general-purpose 8-bit microprocessor family is now available, along with plans for its most popular support circuits, from National Semiconductor.

The National microprocessor, which the company calls the INS8080A, is a direct pin-for-pin and function-for-function replacement for the Intel 8080A device.

The system designer who uses National's 8080A family can take full advantage of the device's many important features, including high performance. Instruction cycle time is better than 2 microseconds.

The device offers powerful programming capability, with 72 problem-solving instructions and multiple register pairs for general-purpose operation. Addressing capability is broad—the 8080A can address up to 65K of memory and up to 512 input-output ports.

Operation of the 8080A is easily interrupted because the program counter is automatically saved during the interruption. Also, the system has the ability to provide vectored interrupts.

The 8080A permits controlled suspension of processor operations. This feature is especially useful when the CPU is operating with a low-speed memory, and it provides the system with direct memory access (DMA) input capability.

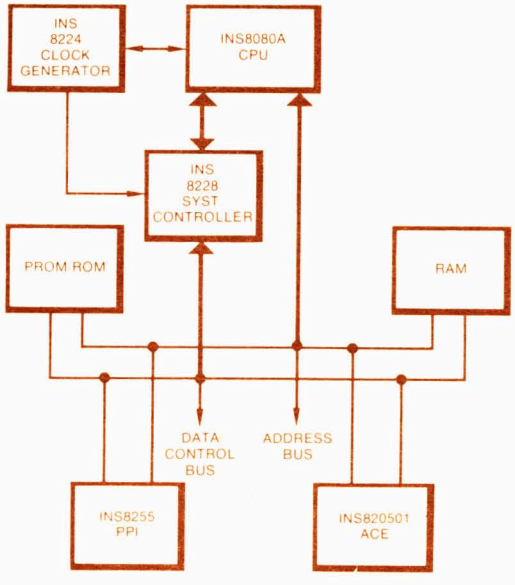
In supporting its 8080A, National is putting its Schottky bipolar technology to work. It is planned for the INS8224 clock generator, which provides timing signals for the CPU and for the system, and the INS8228 provides system control and buffering of the data bus. Further interfacing of bus and control lines can be implemented with National's wide variety of linear and digital interface components.

National's 8080A family line-up permits design of very flexible systems. The designer can satisfy system requirements from the wide range of National's RAM, PROM, ROM, and I/O components. Programmable I/O and peripheral functions allow the designer the freedom to configure and adapt interface lines to his own requirements.

For program storage, the flexible PROM is matched by direct replacement ROMs. For random-access storage, the designer can choose from one of the industry's broadest lines of RAMs which encompass bipolar, MOS, and CMOS technologies.

From experimentation to final production, National hardware and software support is planned to provide the full range of development tools, from basic design kits and easy-to-implement development systems to a full complement of cross and resident assemblers.

Sample quantities of National's 8080A family components are now available from factory stock, and orders are being accepted through the company's distributors and sales representatives. When ordered in lots of 100, the price of the CPU chip is $19.95 each. In 1977, the price in volumes greater than 10,000 will be less than $12 each.




***Block Diagram of National "8080A" Microprocessor System***

(Continued on page 2)

(Continued from page 1)

## N8080 Family

| Part No. | Description | Availability |
|---|---|---|
| INS8080A | 8-BIT CPU, 2$\mu$ SEC CYCLE | NOW |
| INS8224 | CLOCK GENERATOR | NOW |
| INS8228 | SYSTEM CONTROLLER | NOW |
| INS8212 | 8-BIT I/O PORT | NOW |
| INS8255 | PROGRAMABLE PERIPHERAL INTERFACE | NOW |
| INS82501 | ASYNCHRONOUS COMMUNICATION ELEMENT | NOW |

## N8080 Support Devices

| Part No. | Description | Availability |
|---|---|---|
| 74LS138 | 1 OF 8 BINARY DECODER | NOW |
| DS8833 | BI DIRECTIONAL BUS DRIVER NON-INV. | NOW |
| DS8835 | BI DIRECTIONAL BUS DRIVER INV. | NOW |
| MM1702A | 256 x 8 PROM, ERASABLE | NOW |
| MM5204Q | 512 x 8 PROM, ERASABLE | NOW |
| DM87S295,6 | 512 x 8 PROM, 100ns | NOW |
| MM5213 | 256 x 8 ROM | NOW |
| MM5242 | 1K x 8 ROM | NOW |
| MM2316A | 2K x 8 ROM | NOW |
| MM74C920 | 256 x 4 STATIC CMOS RAM | NOW |
| MM2101-2 | 256 x 4 STATIC RAM | NOW |
| MM2111-2 | 256 x 4 STATIC, COMMON I/O | NOW |
| MM2102-2 | 1K x 1 STATIC RAM 650ns | NOW |
| MM2102A-4 | 1K x 1 STATIC RAM 450ns | NOW |
| MM5255,6 | 1K x 4 STATIC RAM | NOW |
| MM5257 | 4K x 1 STATIC RAM | NOW |
| MM5280B | 4K x 1 DYNAMIC RAM | NOW |

# BACK BY POPULAR DEMAND:

# SC/MP!

**by Dave Graves**
Reprinted from the Bit•Bucket, Oct., 1975.

For the designer or user who needs a microprocessor that is powerful, versatile, and inexpensive, National introduces SC/MP (Simple Cost-effective MicroProcessor).

SC/MP represents a significant breakthrough in low-cost computer systems. Providing many of the features of higher-priced systems, SC/MP has sufficient hardware to serve most controller and switching applications where processing speed is not a critical factor. With read/write memory, read-only memory, power supply, chassis, and console, SC/MP becomes a stand-alone microcomputer.

A partial listing of SC/MP architectural features, for example, show why this new CPU chip is so easy to use:

- 46 instructions
- Bidirectional 8-bit data bus
- 16-bit address bus
- Separate serial-data input and output ports
- On-chip oscillator and timing generator—all the user needs is an external capacitor or crystal
- Direct interfacing to standard memories
- On-chip generation of asynchronous control signals for interfacing and a capability of using memories of any speed
- Single 12-volt power supply operation
- Capable of addressing up to 65K bytes of memory
- Two sense ports
- Program interrupt with software enable/disable
- Four, 16-bit address pointer registers—usable by any memory reference instruction for index and displacement, or as software stack pointers
- Multiprocessor network operation—Enable In, Enable Out and Bus Request signals allow direct
- Simple memory addressing—twelve latched addresses, that can directly address up to 4K bytes of standard memory; expandable to 65K bytes simply by latching the four most significant bits
- Start/stop control separate from Reset—allows single-cycle instruction control
- Multiple addressing modes—program counter relative, indexed, auto-indexed and immediate
- Programmed delay—a single instruction controls a 26 $\mu$s to 263 ms delay or time-out signal
- Three user dedicated flags
- 8-bit accumulator and 8-bit extension register
- 8-bit status register—contains the arithmetic carry, overflow, and Interrupt Enable flags; available under program control are two input control flags and three output flags

The SC/MP block diagram on the following page shows how the main components of the chip interface.

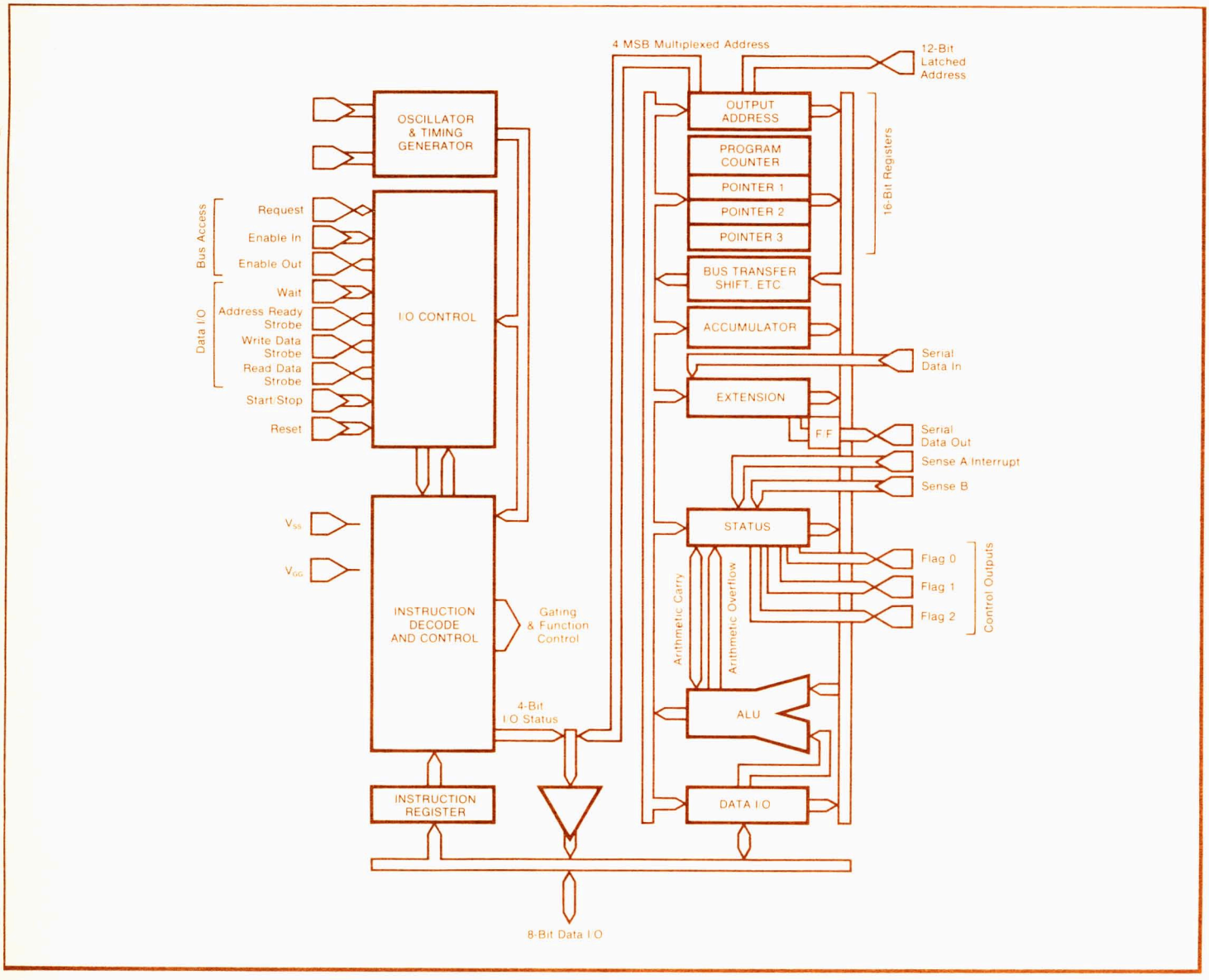
As you can see by the block diagram, SC/MP is an 8-bit parallel processor with 16-bit memory and peripheral device addressing. Functionally, SC/MP has a bidirectional data bus connecting the CPU, memory, and peripheral devices. Peripheral devices are assigned memory addresses, and any standard memory reference instruction can be used for input/output operations.

SC/MP is the first microprocessor designed to fit the immense variety of applications in which 4-bit microprocessors are too difficult to use and for which currently available 8-bit microprocessors are too powerful and expensive—applications that involve low-speed man/machine interfaces in the industrial/commercial and consumer marketplaces.

For example, in the industrial/commercial area, SC/MP is particularly suitable for electronic cash registers, traffic light controllers, elevator controllers, automatic computing-type price/weight scales, measurement and instrument controllers, and word processors.

On the consumer front, SC/MP is ideal for sophisticated calculators, electronic games, appliance controls, home air conditioning/heating and security systems, automatic tuners for TV receivers and mobil communication systems.

SC/MP, a chip for all seasons!




# SC/MP KIT

A completely functional 8-bit microcomputer system, based on the SC/MP microprocessor, can be assembled in less than 50 minutes with National's new SC/MP KIT.

The SC/MP KIT includes all the firmware and components that a person needs to build a working system. Priced at $99 each in quantities up to ten, the kit contains the following components:

• One SC/MP Microprocessor (model ISP-8A/500D) —an 8-bit single-chip central processing unit housed in a 40-pin dual-in-line ceramic package. The SC/MP features static operation; forty-six instruction types; single-byte and double-byte operation; software controlled interrupt structure; built-in serial input-and-output ports; bidirectional 8-bit TRI-STATE® parallel data port; and a latched 12-bit TRI-STATE® address port.

• One Read-Only Memory (model MM5204)—a 4,096-bit ROM organized into 512 bytes, with 8 bits-per-byte. It is pre-programmed to contain KITBUG, which is a monitor and debugging program that assists in the development of the user's application programs. KITBUG provides teletypewriter input-and-output routines and allows examination, modification, and controlled execution of the user's programs.

• Two 1K Random-Access Memories (model MM2101N) —these two RAMs are organized into 256 four-bit words. Together, they provide 256 eight-bit bytes of static read-and-write memory for storage of the user's application programs. The transfer of data to and from the RAM section is controlled by the SC/MP microprocessor and the "KITBUG" program.

• One Voltage Regulator (model LM320MP-12)—this regulator provides a stable -7 volt supply for the microprocessor chip, eliminating the need for an extra power supply.

• One 8-bit Data Buffer (model DM81LS95N)—this buffer provides the interface between the memory and the SC/MP microprocessor's data lines.

(Continued on page 4)

SC/MP Continued

- One Timing Crystal—provides a 1.000-megahertz timing signal for the clock circuit on the SC/MP microprocessor chip. This is the only external timing component needed by the clock.
- One Teletype Interface (model DM7414N)—this IC provides buffer and drive capabilities to implement a 20-millampere current loop interface for a teletypewriter.
- One 72-pin Edge-Connector—this standard connector simplifies interconnection between the SC/MP KIT board and external hardware.
- One 24-pin IC Socket—for easy mounting of the MM5204Q ROM.
- One 40-pin IC Socket—for quickly mounting the SC/MP microprocessor onto the circuit board.
- One Printed Circuit Board—this 4 x 5-inch (10 x 13-cm) PC board provides all component interconnections. It simplifies assembly of the kit and reduces the possibility of assembly errors.
- Eight capacitors.
- Seven resistors.

Once the kit has been assembled, the user can explore the capabilities of the SC/MP microprocessor. While the ROM contains the KITBUG monitor program for the user's convenience, he or she can add other ROMs or PROMs with different programs. The KITBUG firmware lets the user enter programs directly into the read-write memory from a teletype keyboard. The user can then execute the program while examining the contents of the memory and the SC/MP registers to monitor the program's performance.

The SC/MP TTL-compatible input-output interface simplifies the connection of the user's application hardware so that he or she can easily implement practical "real-life" test and demonstration circuits. With SC/MP control-oriented instruction set, the hardware is controlled by the user's application programs.

SC/MP KIT offers users a cheap and easy way to bring a basic system up and they are available now!!!

# PROTOTYPING BOARD

There is a P.C. prototyping board available for users building SC/MP or PACE microprocessors. The boards cost $18.00 each, and have the following features:

- Standard 72-pin card—4-3/8" x 4-7/8".
- Accepts sockets for 14, 16, 24 or 40-pin packages.
- Accepts a maximum of twenty-six 16-pin plus two 14-pin sockets.
- Four additional 16-pin sockets used as wire/wrap pins for edge connector.
- Spacing included for 22-pin packages.
- Holes for 50-pin 3M flat cable socket.
- $V_{CC}$ and GND bussing includes traces for filter caps between each socket.

Contact: Norm Inskeep
2185 Conway Street
Milpitas, CA 95035
(408) 263-4065

# MICROPROCESSOR CONSULTANT & PRODUCT ANNOUNCEMENT

**ABLER DATA SERVICE INC.**
**740 GARVENS AVE.**
**BROOKFIELD, WISCONSIN 53005**
**(414) 786-2448**

Dr. Joseph H. Abler has been working with IMP, PACE & SC/MP applications in the Wisconsin and Illinois area. His company has developed a Digital I/O card for PACE application card users. One of the applications he has used PACE for was a concrete batching plant. If any of you would like reprints of the article describing this system please contact him at the address above.

## PRODUCT ANNOUNCEMENT

**Digital I/O Interface Card**—ADS-P001 **Price $145.00**

This circuit is designed to interface directly with the PACE application CPU card. Some of the features are:

—72 contact pinout on 4.375 by 4.862 inch card

—16 bit TTL input. Word addressable via LD instruction.

```
e. g.  LD   R0   , @ADDR   ; LOAD 16 BITS INTO R0
         .
         .
ADDR:  .WORD  081NX   ; WHERE N IS HEX ADDRESS
                      ; ON THE CIRCUIT DIP
                        SWITCH.
                      ; X IS A DON'T CARE
```

—8 bit TTL output with complements. They are latched and bit addressable. Bit 0 of the output register is used.

```
ST   R0,  @ADDR   ; BIT 0 OF R0 IS LATCHED.
                  ; WHERE N IS AS ABOVE.
                  ; X IS A HEX ADDRESS FROM
                  ; 0 TO 7 TO SELECT ONE OF
                  ; EIGHT LATCHES.
```

ADS-P002

**Diagnostic extender card for the above card.** **Price $40.00**

This card provides I/O interconnection for automatic software trouble-shooting.

ADS-P003

**Diagnostic software to test Digital I/O Card.** **Price $60.00**

This program provides automatic testing and bad chip detection. DOCUMENTATION INCLUDED.

# NEW PROGRAMS

The following new programs (source listing only available) are updates to PROMSFT that allow a programmer to program proms from the disc.

SL0023A DISC RLM FOR PRMSFT-B
SL0024A DISC RLM FOR PRMSFT-C

NATIONAL SEMICONDUCTOR
IMP-16
RESIDENT ASSEMBLER

```
TITLE:  DRLM    DISC RLM FOR PRMSFT-B

DRLM   DISC RLM FOR PRMSFT-B


  1               ;       PATCH FOR PRMSFT REV  B
  2               ;       WRITTEN BY BARNEY HORDOS
  3               ;
  4               ;
  5               ; TO INCORPORATE THIS PATCH IN PRMSFT REV  B ON A DIS
  6               ;       INITIALIZE AND LOAD DISC
  7               ;       SET PC TO X'C000
  8               ;       SET AC0 TO X'006C
  9               ;       PUSH RUN TTY WILL TYPE:
 10               ;               DISC LOADER (REV  E) READY
 11               ;               '
 12               ;       TYPE LM (CR)
 13               ;       TYPE MP 0D4 (CR)
 14               ;       TYPE OBS 0 (CR)
 15               ;       LOAD PRMSFT REV  B TAPE (4360343-A)
 16               ;       TYPE RLM (CR)
 17               ;       TURN ON READER TTY WILL TYPE:
 18               ;               PRMSFT P00343B 03/19/75  XXXX  XXXX
 19               ;               BS=0000 0039 AS=0118 08B1 ENT 0300
 20               ;       TURN READER OFF
 21               ;       LOAD PATCH TAPE
 22               ;       TYPE RLM (CR) TTY WILL TYPE:
 23               ;               DRLM DISC RLM FOR PRMSFT-B  XXXX  XXX
 24               ;               BS=003A 003A AS=0341 09A0 ENT=0300
 25               ;       TURN READER OFF
 26               ;       TYPE GO (CR)
 27               ;       PROGRAM IS WRITTEN TO DISC TTY WILL TYPE
 28               ;  BS=0000 003A AS=0118 09A0 PTR=0100 0100 ENT=0300
 29               ;   PGM WRITTEN TO SECTORS 00D4 00DC
 30               ;
 31               ;       TO EXECUTE PRMSFT
 32               ;       INITIALIZE
 33               ;       SET PC TO X'C000
 34               ;       SET AC0 TO X'00D4
 35               ;       RUSH RUN
 36               ;       WHEN PROGRAM IS READ IN TTY WILL TYPE
 37               ;               PRMSFTB
 38               ;               ?
 39               ;       U (CR) WILL NOW READ AN LM FROM DISC
 40               ;
 41               ;       EXAMPLE OF READING LM FROM DISC TO X'1000
 42               ;               ? 0 1000 (CR)
 43               ;               1000
 44               ;               ? U (CR)
 45               ;               SECTOR NUMBER IN HEX = 220 (CR)
 46               ;               FIRST ADDRESS = XXXX
 47               ;               DONE
 48               ;               ? M 1000 (CR)
 49               ;               LEFT BYTE TYPE L(CR)  RIGH BYTE TYPE
 50               ;               L (CR)
 51               ;               DONE
 52               ;       NOW PROGRAM PROM
 53
 54                       TITLE DRLM,' DISC RLM FOR PRMSFT-B
 55      0000             ASECT
 56      0900             =       0900
 57
 58      0000    BSTART   =       0
 59      0033    BUFFAD   =       033
 60      0038    FIRAD    =       038
 61      0005    NZRO     =       5
 62      0036    K3FFF    =       036
 63      0001    ZERO     =       1
 64      0011    MESSA    =       011
 65      07C7    CKSMER   =       07C7            ;CKER
 66      0F00    DIS      =       0F00
 67      0027    CRLFA    =       027             ;CRLF
 68      0019    WHAT     =       019
 69      0002    GTEO     =       2
 70
 71
 72              ;       COMMANDS
 73
 74              ;  U CR  READ DISC RLM
 75
 76              ;  CTRL 0 ABORTS COMMAND
 77
 78              SECTN
 79 0900 2D34 A          JSR     @NEWA           ;FIX BUFFER
 80 0901 A114 A          ST      0,ANDD
 81 0902 2C11 A          JSR     @MESSA
 82 0903 0927 A           WORD   SECT
 83 0904 2D33 A          JSR     @GETWA          ;GET SECTOR
 84 0905 2C27 A          JSR     @CRLFA
 85 0906 8019 A          LD      0,WHAT          ;GET LAST WORD
 86 0907 0200 A          RTS
 87
 88                      DRLM
 89 0908 29F7 A          JSR     SECTN           ;GET SECTOR NUMBER
 90 0909 A161 A          ST      0,SEC           ;SAVE SECTOR NUMBER
 91 090A 295A A          JSR     DISC            ;READ FROM DISC
 92 090B 8D60 A          LD      3,DBUF
 93                      LLD
 94 090C BC00 I          ST      3,DSTART
 95 090D 8300 A          LD      0,(3)           ;GET FIRST WORD
 96 090E 1216 A          BOC     GTEO,DTORS      ;GO IF TITLE OR SYMBO
 97 090F 7127 A          SKAZ    0,D4000         ;CHECK FOR END RECORD
 98 0910 2525 A          JMP     @ENDRA
 99                      DDATA
100 0911 2927 A          JSR     DCKSM           ;CHECK SUM RECORD
101 0912 9C00 I          LD      3,DSTART
102 0913 8700 A          LD      1,(3)           ;GET LENGTH
103 0914 6436 A          AND     1,K3FFF
104 0915 8303 A          LD      0,3(3)          ;GET STARTING ADDRESS
105 0916 0000 A ANDD      WORD   0
106 0917 3281 A          RCPY    0,2             ;ADDRESS OF TEMP BUFF
107 0918 C833 A          ADD     2,BUFFAD
108 0919 8038 A          LD      0,FIRAD         ;CHECK IF FIRST DATA
109 091A 1501 A          BOC     NZRO, +2
110 091B A838 A          ST      2,FIRAD
111 091C 49FC A          AISZ    1,-4
112 091D 4B06 A          AISZ    3,6             ;UPDATE POINTER
113                      DCRST
114 091E 8300 A          LD      0,(3)           ;GET DATA WORD
115 091F A200 A          ST      0,(2)           ;STORE WORD
116 0920 4B01 A          AISZ    3,1
117 0921 4A01 A          AISZ    2,1
118 0922 49FF A          AISZ    1,-1
119 0923 21FA A          JMP     DCRST
120 0924 21E7 A          JMP     LLD
121
122                      DTORS:
123 0925 2913 A          JSR     DCKSM           ;CHECK SUM
124 0926 21E5 A          JMP     LLD
125
126 0927 0D0A A SECT      WORD   0D0A
127 0928 5345 A           ASCII  'SECTOR NUMBER IN HEX = '
    0929 4354 A
    092A 4F52 A
    092B 204E A
    092C 554D A
    092D 4245 A
    092E 5220 A
    092F 494E A
    0930 2048 A
    0931 4558 A
    0932 203D A
    0933 2020 A
128 0934 0000 A           WORD   0
129
130 0935 060C A NEWA      WORD   060C            ;NEW
131 0936 0639 A ENDRA     WORD   0639            ;PTNGI
132 0937 4000 A D4000     WORD   04000
133 0938 035B A GETWA     WORD   035B            ;START
134
135
136                      DCKSM:
137 0939 8D65 A          LD      3,DSTART        ;GET STARTING ADDRESS
138 093A 8300 A          LD      0,(3)           ;GET LENGTH
139 093B 6036 A          AND     0,K3FFF
140 093C 3181 A          RCPY    0,1             ;SAVE LENGTH IN AC1
141 093D 8301 A          LD      0,1(3)          ;GET CHECK SUM
142 093E 1120 A          BOC     ZERO,NOCKSM
143 093F 5001 A          CAI     0,1             ;FORM 2'S COMP
144 0940 4B02 A          AISZ    3,2             ;UPDATE POINTER
145                      CL:
146 0941 C300 A          ADD     0,(3)           ;MAKE CKSM
147 0942 4B01 A          AISZ    3,1
148 0943 290A A          JSR     CDON            ;CHECK IF ENOUGH BUFF
149 0944 49FF A          AISZ    1,-1            ;BUMP LENGTH
150 0945 21FB A          JMP     CL
151                      OUT:
152 0946 4B02 A          AISZ    3,2             ;CHECK IF LENGTH IS 0
153 0947 2906 A          JSR     CDON
154 0948 1103 A          BOC     ZERO,OK         ;CHECK IF CKSM IS ZER
155 0949 2C11 A          JSR     @MESSA
156 094A 07C7 A           WORD   CKSMER
157 094B 2000 A          JMP     0
158                      OK:
159 094C 4BFE A          AISZ    3,-2            ;FIX POINTER
160 094D 0200 A          RTS
161
162                      CDON:
163 094E ED51 A          SKG     3,DEND          ;CHECK IF END OF BUFF
164 094F 0200 A          RTS
165 0950 894F A          LD      2,DEND
166 0951 5201 A          CAI     2,1             ;2'S COMP
167 0952 C94C A          ADD     2,DSTART        ;MAKE -DIFFERENCE
168 0953 C918 A          ADD     2,DBUF          ;NEW STARTING ADDRESS
169 0954 8D4A A          LD      3,DSTART
170 0955 A949 A          ST      2,DSTART
171
172                      ML:
173 0956 8300 A          LD      0,(3)           ;GET WORD TO MOVE
174 0957 A200 A          ST      0,(2)           ;MOVE IT
175 0958 4A01 A          AISZ    2,1
176 0959 4B01 A          AISZ    3,1
177 095A E911 A          SKG     2,DBUF          ;CHECK IF DONE
178 095B 21FA A          JMP     ML              ;NO
179 095C 2908 A          JSR     DISC            ;READ NEXT SECTOR
180 095D 4400 A          PULL    0
181 095E 21DA A          JMP     DCKSM           ;REDO CKSM
182
183                      NOCKSM
184 095F 4B02 A          AISZ    3,2             ;FIX POINTER
185 0960 4B01 A          AISZ    3,1
186 0961 29EC A          JSR     CDON
187 0962 49FF A          AISZ    1,-1
188 0963 21FB A          JMP     NOCKSM
189 0964 21E1 A          JMP     OUT
190
191                      DISC
192 0965 2D38 A          JSR     @DISCIO
193 0966 096A A           WORD   READD
194 0967 2107 A          JMP     DER
195 0968 7902 A          ISZ     SEC
196 0969 0200 A          RTS
197 096A 0002 A READD     WORD   2                READ
198 096B 0000 A SEC       WORD   0
199 096C 0F00 A DBUF      WORD   DIS
```

```
 200 096D 0000 A  DSTAT:   WORD   0
 201 096E 0000 A  PSECT:   WORD   0
 202
 203
 204              DER:
 205 096F 2C11 A           JSR    @MESSA
 206 0970 0979 A           WORD   ER
 207 0971 81FB A           LD     0,DSTAT          ;GET ERROR STATUS
 208 0972 7128 A           SKAZ   0,D80            ;CHECK FOR READ ERROR
 209 0973 211E A           JMP    DRDER
 210 0974 7127 A           SKAZ   0,D1             ;CHECK FOR MISSING SY
 211 0975 211F A           JMP    MISYNC
 212 0976 7126 A           SKAZ   0,D8             ;CHECK FOR NOT ON LIN
 213 0977 2120 A           JMP    NOTRD
 214 0978 2000 A           JMP    BSTART
 215
 216 0979 0D0A A  ER        WORD   0D0A
 217 097A 4449 A           ASCII  'DISC ERROR  '
     097B 5343 A
     097C 2045 A
     097D 5252 A
     097E 4F52 A
     097F 2E20 A
 218 0980 0000 A           WORD   0
 219              DRR:
 220 0981 4F4E A           ASCII  'ON READ '
     0982 2052 A
     0983 4541 A
     0984 4420 A
 221 0985 0000 A           WORD   0
 222
 223              SYNC:
 224 0986 4F4F A           ASCII  'NO SYNC '
     0987 2053 A
     0988 594E A
     0989 4320 A
 225 098A 0000 A           WORD   0
 226
 227              NRDY:
 228 098B 4E4F A           ASCII  'NOT ON LINE'
     098C 5420 A
     098D 4F4E A
     098E 204C A
     098F 494E A
     0990 4520 A
 229 0991 0000 A           WORD   0
 230
 231
 232
 233              DRDER:
 234 0992 2C11 A           JSR    @MESSA
 235 0993 0981 A           WORD   DRR
 236 0994 2000 A           JMP    BSTART
 237
 238              MISYNC:
 239 0995 2C11 A           JSR    @MESSA
 240 0996 0986 A           WORD   SYNC
 241 0997 2000 A           JMP    BSTART
 242
 243              NOTRD:
 244 0998 2C11 A           JSR    @MESSA
 245 0999 098B A           WORD   NRDY
 246 099A 2000 A           JMP    BSTART
 247
 248
 249 099B 0080 A  D80       WORD   080
 250 099C 0001 A  D1        WORD   1
 251 099D 0008 A  D8        WORD   8
 252 099E C008 A  DISCIO    WORD   0C008
 253 099F 0000 A  BSTART    WORD   0
 254 09A0 1000 A  DEND      WORD   DIS+256
 255
 256       06F9             =      06F9
 257 06F9 0908 A            WORD   DRLM             ; U COMMAND DISC RLM
 258
 259       06D9             =      06D9            ;PROTEC
 260 06D9 09A0 A            WORD   DEND             ;END OF PROTECTED
 261
 262       0341             =      0341
 263 0341 0011 A            WORD   011              ;CTRL Q FOR ABORT
 264
 265
 266       0300             END    0300            ;START

     0000 099F A
ANDD     0916  A       BSTART  0000  A       BUFFAD  0033  A
CDON     094E  A       CKSMER  07C7  A       CI      0941  A
CRLFA    0027  A       D1      099C  A       D4000   0937  A
D8       099D  A       D80     099B  A       DBUF    0960  A
DCKSM    0939  A       DCRST   091E  A       DDATA   0911  A*
DEND     09A0  A       DER     096F  A       DIS     0F00  A
DISC     0965  A       DISCIO  099E  A       DRDER   0992  A
DRLM     0908  A       DRR     0981  A       DSTART  099F  A
DSTAT    096D  A       DTORS   0925  A       ENDRA   0936  A
ER       0979  A       FIRAD   0038  A       GETWA   0938  A
GTEO     0002  A       K3FFF   0036  A       LLD     090C  A
MESSA    0011  A       MISYNC  0995  A       ML      0956  A
NEWA     0935  A       NOCKSM  095F  A       NOTRD   0998  A
NRDY     098B  A       NZRO    0005  A       OK      094C  A
OUT      0946  A       PSECT   096E  A*      READD   096A  A
SEC      096B  A       SECT    0927  A       SECTN   0900  A
SYNC     0986  A       WHAT    0019  A       ZERO    0001  A
```

## TITLE: DRLM  DISC RLM FOR PRMSFT-C

```
DRLM   DISC RLM FOR PRMSFT-C



  1              ; TO INCORPORATE THIS PATCH IN PRMSFT REV C ON A DISC
  2              ;       INITIALIZE
  3              ;       SET PC TO X'C000
  4              ;       SET AC0 TO X'006C
  5              ;       PUSH RUN TTY WILL TYPE
  6              ;               DISC LOADER (REV  E) READY
  7              ;               *
  8              ;       TYPE LM (CR)
  9              ;       TYPE MP 0D4 (CR)
 10              ;       TYPE OBS 0 (CR)
 11              ;       LOAD PRMSFT REV C TAPE (4360343-A)
 12              ;       TYPE RLM (CR)
 13              ;       TURN ON READER TTY WILL TYPE:
 14              ;               PRMSFT P00343C 12/03/75  XXXX  XXXX
 15              ;               BS=0000:0039 AS=0118:08B1 ENT 0300
 16              ;       TURN READER OFF
 17              ;       LOAD PATCH TAPE
 18              ;       TYPE RLM (CR) TTY WILL TYPE:
 19              ;               DRLM DISC RLM FOR PRMSFT-C  XXXX  XXX
 20              ;               BS=003A:003A AS=0341:09A0 ENT=0300
 21              ;       TURN READER OFF
 22              ;       TYPE GO (CR)
 23              ;       PROGRAM IS WRITTEN TO DISC TTY WILL TYPE:
 24              ;  BS=0000:003A AS=0118:09A0 PTR=0100:0100 ENT=0300
 25              ;   PGM WRITTEN TO SECTORS 00D4:00DC
 26
 27
 28              ;       TO EXECUTE PRMSFT:
 29              ;       INITIALIZE
 30              ;       SET PC TO X'C000
 31              ;       SET AC0 TO X'00D4
 32              ;       PUSH RUN
 33              ;       WHEN PROGRAM IS READ IN TTY WILL TYPE:
 34              ;               PRMSFTC
 35              ;               ?
 36              ;       U (CR) WILL NOW READ AN LM FROM DISC
 37
 38
 39              ;       EXAMPLE OF READING LM FROM DISC TO X'1000:
 40              ;               ? 0 1000 (CR)
 41              ;               1000
 42              ;               ? U (CR)
 43              ;               SECTOR NUMBER IN HEX = 220 (CR)
 44              ;               FIRST ADDRESS = XXXX
 45              ;               DONE
 46              ;               ? M 1000 (CR)
 47              ;               LEFT BYTE TYPE L(CR)  RIGH BYTE TYPE
 48              ;               L (CR)
 49              ;               DONE
 50              ;       NOW PROGRAM FROM
 51
 52
 53
 54                       .TITLE DRLM,' DISC RLM FOR PRMSFT-C '
 55       0000            .ASECT
 56       0900            .       =       0900
 57
 58       0000    BSTART  =       0
 59       0033    BUFFAD  =       033
 60       0038    FIRAD   =       038
 61       0005    NZRO    =       5
 62       0036    K3FFF   =       036
 63       0001    ZERO    =       1
 64       0011    MESSA   =       011
 65       07CA    CKSMER  =       07CA
 66       0F00    DIS     =       0F00
 67       0027    CRLFA   =       027             ;CRLF
 68       0019    WHAT    =       019
 69       0002    GTEO    =       2
 70
 71
 72                       ;       COMMANDS
 73
 74                       ;  U CR  READ DISC RLM
 75
 76                       ;  CTRL Q ABORTS COMMAND
 77
 78                       SECTN:
 79 0900 2D34 A                    JSR     @NEWA           ;FIX BUFFER.
 80 0901 A114 A                    ST      0,ANDD
 81 0902 2C11 A                    JSR     @MESSA
 82 0903 0927 A                     WORD   SECT
 83 0904 2D33 A                    JSR     @GETWA          ;GET SECTOR.
 84 0905 2C27 A                    JSR     @CRLFA
 85 0906 8019 A                    LD      0,WHAT          ;GET LAST WORD.
 86 0907 0200 A                    RTS
 87
 88                       DRLM:
 89 0908 29F7 A                    JSR     SECTN           ;GET SECTOR NUMBER.
 90 0909 A161 A                    ST      0,SEC           ;SAVE SECTOR NUMBER.
 91 090A 295A A                    JSR     DISC            ;READ FROM DISC.
 92 090B 8D60 A                    LD      3,DBUF
 93                       LLD:
 94 090C BC00 I                    ST      3,DSTART
 95 090D 8300 A                    LD      0,(3)           ;GET FIRST WORD.
 96 090E 1216 A                    BOC     GTEO,DTORS      ;GO IF TITLE OR SYMBO
 97 090F 7127 A                    SKAZ    0,D4000         ;CHECK FOR END RECORD
 98 0910 2525 A                    JMP     @ENDRA
 99                       DDATA:
100 0911 2927 A                    JSR     DCKSM           ;CHECK SUM RECORD.
101 0912 9C00 I                    LD      3,DSTART
102 0913 8700 A                    LD      1,(3)           ;GET LENGTH
103 0914 6436 A                    AND     1,K3FFF
104 0915 8303 A                    LD      0,3(3)          ;GET STARTING ADDRESS
105 0916 0000 A  ANDD:              WORD    0
106 0917 3281 A                    RCPY    0,2             ;ADDRESS OF TEMP BUFF
107 0918 C833 A                    ADD     2,BUFFAD
108 0919 8038 A                    LD      0,FIRAD         ;CHECK IF FIRST DATA
109 091A 1501 A                    BOC     NZRO, +2
110 091B A838 A                    ST      2,FIRAD
111 091C 49FC A                    AISZ    1,-4
112 091D 4B06 A                    AISZ    3,6             ;UPDATE POINTER.
113                      DCRST:
114 091E 8300 A                    LD      0,(3)           ;GET DATA WORD.
115 091F A200 A                    ST      0,(2)           ;STORE WORD.
116 0920 4B01 A                    AISZ    3,1
117 0921 4A01 A                    AISZ    2,1
118 0922 49FF A                    AISZ    1,-1
119 0923 21FA A                    JMP     DCRST
120 0924 21E7 A                    JMP     LLD
121
122                      DTORS:
123 0925 2913 A                    JSR     DCKSM           ;CHECK SUM.
124 0926 21E5 A                    JMP     LLD
125
126 0927 0D0A A  SECT:     .WORD   0D0A
127 0928 5345 A            .ASCII  'SECTOR NUMBER IN HEX = '
    0929 4354 A
    092A 4F52 A
    092B 204E A
    092C 554D A
    092D 4245 A
```

```
    092E 5220 A
    092F 494E A
    0930 2048 A
    0931 4558 A
    0932 203D A
    0933 2020 A
128 0934 0000 A             .WORD   0
129
130 0935 060F A  NEWA:      .WORD   060F            ;NEW
131 0936 063C A  ENDPA:     .WORD   063C            ;PTNGI
132 0937 4000 A  D4000:     .WORD   04000
133 0938 035B A  GETWA:     .WORD   035B            ;START
134
135
136                 DCKSM:
137 0939 8D65 A             LD      3,DSTART        ;GET STARTING ADDRESS
138 093A 8300 A             LD      0,(3)           ;GET LENGTH
139 093B 6036 A             AND     0,K3FFF
140 093C 3181 A             RCPY    0,1             ;SAVE LENGTH IN AC1
141 093D 8301 A             LD      0,1(3)          ;GET CHECK SUM
142 093E 1120 A             BOC     ZERO,NOCKSM
143 093F 5001 A             CAI     0,1             ;FORM 2'S COMP
144 0940 4B02 A             AISZ    3,2             ;UPDATE POINTER
145                 CL:
146 0941 C300 A             ADD     0,(3)           ;MAKE CKSM
147 0942 4B01 A             AISZ    3,1
148 0943 290A A             JSR     CDON            ;CHECK IF ENOUGH BUFF
149 0944 49FF A             AISZ    1,-1            ;BUMP LENGTH
150 0945 21FB A             JMP     CL
151                 OUT:
152 0946 4B02 A             AISZ    3,2             ;CHECK IF LENGTH IS 0
153 0947 2906 A             JSR     CDON
154 0948 1103 A             BOC     ZERO,OK         ;CHECK IF CKSM IS ZER
155 0949 2C11 A             JSR     @MESSA
156 094A 07CA A             .WORD   CKSMER
157 094B 2000 A             JMP     0
158                 OK:
159 094C 4BFE A             AISZ    3,-2            ;FIX POINTER
160 094D 0200 A             RTS
161
162                 CDON:
163 094E ED51 A             SKG     3,DEND          ;CHECK IF END OF BUFF
164 094F 0200 A             RTS
165 0950 894F A             LD      2,DEND
166 0951 5201 A             CAI     2,1             ;2'S COMP
167 0952 C94C A             ADD     2,DSTART        ;MAKE -DIFFERENCE
168 0953 C918 A             ADD     2,DBUF          ;NEW STARTING ADDRESS
169 0954 8D4A A             LD      3,DSTART
170 0955 A949 A             ST      2,DSTART
171
172                 ML:
173 0956 8300 A             LD      0,(3)           ;GET WORD TO MOVE
174 0957 A200 A             ST      0,(2)           ;MOVE IT
175 0958 4A01 A             AISZ    2,1
176 0959 4B01 A             AISZ    3,1
177 095A E911 A             SKG     2,DBUF          ;CHECK IF DONE
178 095B 21FA A             JMP     ML              ;NO
179 095C 2908 A             JSR     DISC            ;READ NEXT SECTOR
180 095D 4400 A             PULL    0
181 095E 21DA A             JMP     DCKSM           ;REDO CKSM
182
183                 NOCKSM:
184 095F 4B02 A             AISZ    3,2             ;FIX POINTER
185 0960 4B01 A             AISZ    3,1
186 0961 29EC A             JSR     CDON
187 0962 49FF A             AISZ    1,-1
188 0963 21FB A             JMP     NOCKSM
189 0964 21E1 A             JMP     OUT
190
191                 DISC:
192 0965 2D38 A             JSR     @DISCIO
193 0966 096A A             .WORD   READD
194 0967 2107 A             JMP     DER
195 0968 7902 A             ISZ     SEC
196 0969 0200 A             RTS
197 096A 0002 A  READD:     .WORD   2               ;READ
198 096B 0000 A  SEC:       .WORD   0
199 096C 0F00 A  DBUF:      .WORD   DIS
200 096D 0000 A  DSTAT:     .WORD   0
201 096E 0000 A  PSECT:     .WORD   0
202
203
204                 DER:
205 096F 2C11 A             JSR     @MESSA
206 0970 0979 A             .WORD   ER
207 0971 81FB A             LD      0,DSTAT         ;GET ERROR STATUS
208 0972 7128 A             SKAZ    0,D80           ;CHECK FOR READ ERROR
209 0973 211E A             JMP     DRDER
210 0974 7127 A             SKAZ    0,D1            ;CHECK FOR MISSING SY
211 0975 211F A             JMP     MISYNC
212 0976 7126 A             SKAZ    0,D8            ;CHECK FOR NOT ON LIN
213 0977 2120 A             JMP     NOTRD
214 0978 2000 A             JMP     BSTART
215
216 0979 0D0A A  ER:        .WORD   0D0A
217 097A 4449 A             .ASCII  'DISC ERROR '
    097B 5343 A
    097C 2045 A
    097D 5252 A
    097E 4F52 A
    097F 2E20 A
218 0980 0000 A             .WORD   0
219                 DRR:
220 0981 4F4E A             .ASCII  'ON READ '
    0982 2052 A
    0983 4541 A
    0984 4420 A
221 0985 0000 A             .WORD   0
222
223                 SYNC:
224 0986 4E4F A             .ASCII  'NO SYNC '
    0987 2053 A
    0988 594E A
    0989 4320 A
225 098A 0000 A             .WORD   0
226
227                 NRDY:
228 098B 4E4F A             .ASCII  'NOT ON LINE '
    098C 5420 A
    098D 4F4E A
    098E 204C A
    098F 494E A
    0990 4520 A
229 0991 0000 A             .WORD   0
```

```
230
231
232
233                 DRDER:
234 0992 2C11 A             JSR     @MESSA
235 0993 0981 A             .WORD   DRR
236 0994 2000 A             JMP     BSTART
237
238                 MISYNC:
239 0995 2C11 A             JSR     @MESSA
240 0996 0986 A             .WORD   SYNC
241 0997 2000 A             JMP     BSTART
242
243                 NOTRD:
244 0998 2C11 A             JSR     @MESSA
245 0999 098B A             .WORD   NRDY
246 099A 2000 A             JMP     BSTART
247
248
249 099B 0080 A  D80:       .WORD   080
250 099C 0001 A  D1:        .WORD   1
251 099D 0008 A  D8:        .WORD   8
252 099E C008 A  DISCIO:    .WORD   0C008
253 099F 0000 A  DSTART:    .WORD   0
254 09A0 1000 A  DEND:      .WORD   DIS+256
255
256                 .=      06FC
257 06FC 0908 A             .WORD   DRLM            ;'U' COMMAND DISC RLM
258
259                 .=      06DC                    ;PROTEC
260 06DC 09A0 A             .WORD   DEND            ;END OF PROTECTED
261
262                 .=      0341
263 0341 0011 A             .WORD   011             ;CTRL Q FOR ABORT
264
265
266                 .END    0300                    ;NSTART
```

```
    0000 099F A
ANDD    0916  A     BSTART  0000  A     BUFFAD  0033  A
CDON    094E  A     CKSMER  07CA  A     CL      0941  A
CRLFA   0027  A     D1      099C  A     D4000   0937  A
D8      099D  A     D80     099B  A     DBUF    096C  A
DCKSM   0939  A     DCRST   091E  A     DDATA   0911  A*
DEND    09A0  A     DER     096F  A     DIS     0F00  A
DISC    0965  A     DISCIO  099E  A     DRDER   0992  A
DRLM    0908  A     DRR     0981  A     DSTART  099F  A
DSTAT   096D  A     DTORS   0925  A     ENDPA   0936  A
ER      0979  A     FIRAD   0028  A     GETWA   0938  A
GTEO    0002  A     K3FFF   0036  A     LLD     090C  A
MESSA   0011  A     MISYNC  0995  A     ML      0956  A
NEWA    0935  A     NOCKSM  095F  A     NOTRD   0998  A
NRDY    098B  A     NZRO    0005  A     OK      094C  A
OUT     0946  A     PSECT   096E  A*    READD   096A  A
SEC     096B  A     SECT    0927  A     SECTN   0900  A
SYNC    0986  A     WHAT    0019  A     ZERO    0001  A
```

```
DRLM   DISC RLM FOR PRMSFT-C

NO ERROR LINES
SOURCE CHECKSUM=0C8A
OBJECT CHECKSUM=5263

***DISC SECTORS USED***

FIRST INPUT SECTOR HEX -  0200
FINAL INPUT SECTOR HEX -  0209

FIRST OBJECT SECTOR HEX - 020A
FINAL OBJECT SECTOR HEX - 020B
```

# IMP-16/CRT DOS

## CAUTION

If you have a DISC operating system and are using CASM and have a CRT connected, you should be aware of the following. Page 1-1 of the DOS Users Manual (Pub. No. 4200077A) describes the areas of memory used by the DISC and I/O Routines. For example, if you have an 8K system, memory locations IFE0 to IFFF are used for DISC I/O Pointers and a Save Area. CASM also uses the top of an 8K system to store the symbol table created during the assembly of a program. Thus a conflict between the CRT I/O information and CASM's symbol table will result. This will cause errors in the program assembly that are not obvious. A fix to this problem is to adjust the address that CASM uses for its symbol table by changing memory location 0FC4 from 01FFF to 01FE0.

# TECHNICAL TRAINING ENROLLMENT FORM

**Student Information**

Name _____

Title _____ Telephone _____

Company _____

Address _____

**Course Selected**

☐ Microprocessor Fundamentals Date _____

☐ IMP-16/PACE Applications Date _____

☐ SC/MP Applications Date _____

☐ Advanced Programming Date _____

**See Schedule of Classes Below.**

For further information, contact the Training Center nearest you.

**Training Center**

☐ Eastern Microprocessor Training Center
National Semiconductor Corporation
1320 South Dixie Highway
Coral Gables, Florida 33146
Tel: (305) 661-7969

☐ Western Microprocessor Training Center
National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, California 95051
Tel: (408) 732-5000, Ext. 7183

☐ Other _____

# SCHEDULE OF MICROPROCESSOR CLASSES

| | EASTERN TRAINING CENTER | WESTERN TRAINING CENTER |
|---|---|---|
| MICROPROCESSOR FUNDAMENTALS | SEPTEMBER 27<br>OCTOBER 1<br>NOVEMBER 8–12 | SEPTEMBER 13–17<br>OCTOBER 25–29 |
| SC/MP APPLICATIONS | OCTOBER 4–8<br>NOVEMBER 15–19 | SEPTEMBER 20–24<br>NOVEMBER 8–12 |
| PACE | OCTOBER 18–22 | JULY 26–30<br>SEPTEMBER 27<br>OCTOBER 11<br>NOVEMBER 1–5 |
| ADVANCED PROGRAMMING | AUGUST 23–27 | OCTOBER 3–8 |

## TRAINING CENTER LOCATIONS

Western Microprocessor Training Center/470
National Semiconductor Corp.
2900 Semiconductor Dr.
Santa Clara, California 95051
Telephone (408) 737-5122

Eastern Microprocessor Training Center
National Semiconductor Corp.
1320 South Dixie Highway, Suite 870
Coral Gables, Florida 33146
Telephone (305) 661-7969

# PACE INSTRUCTION SET

## ALPHANUMERIC SEQUENCE BY HEXADECIMAL

Read down then right.

| Mnemonic Assembler Code | | AC0 | AC1 | AC2 | AC3 | BASE PAGE (XX) | PC REL (XX+PC) | AC2 REL (XX+AC2) | AC3 REL (XX+AC3) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HALT | 0000 | | | | | | | | | | | | | | | | |
| CFR r | | 0400 | 0500 | 0600 | 0700 | | | | | | | | | | | | |
| CRF r | | 0800 | 0900 | 0A00 | 0800 | | | | | | | | | | | | |
| PUSH F | 0C00 | | | | | | | | | | | | | | | | |
| PULL F | 1000 | | | | | | | | | | | | | | | | |
| JSR disp(xr) | | | | | | 14XX | 15XX | 16XX | 17XX | | | | | | | | |
| JMP disp(xr) | | | | | | 18XX | 19XX | 1AXX | 1BXX | | | | | | | | |
| XCHRS r | | 1C00 | 1D00 | 1E00 | 1F00 | | | | | | | | | | | | |
| ROL r,n,l | | 20XX | 21XX | 22XX | 23XX | | | | | | | | | | | | |
| ROR r,n,l | | 24XX | 25XX | 26XX | 27XX | | | | | | | | | | | | |
| SHL r,n,l | | 28XX | 29XX | 2AXX | 2BXX | | | | | | | | | | | | |
| SHR r,n,l | | 2CXX | 2DXX | 2EXX | 2FXX | | | | | | | | | | | | |
| fc | | NOT USED | IE1 | IE2 | IE3 | IE4 | IE5 | OVF | CRY | LINK | IEN | BYTE | F11 | F12 | F13 | F14 | NOT USED |
| PFLG fc | | 3000 | 3100 | 3200 | 3300 | 3400 | 3500 | 3600 | 3700 | 3800 | 3900 | 3A00 | 3B00 | 3C00 | 3D00 | 3E00 | 3F00 |
| SFLG fc | | 3080 | 3180 | 3280 | 3380 | 3480 | 3580 | 3680 | 3780 | 3880 | 3980 | 3A80 | 3B80 | 3C80 | 3D80 | 3E80 | 3F80 |
| cc | | STACK Full | AC0 = 0 | AC0 Bit15=0 | AC0 Bit0=1 | AC0 Bit1=1 | AC0 ≠ 0 | AC0 Bit2=1 | CONT | LINK | IEN | CRY | AC0 Bit15=0 | OVF | JC13 | JC14 | JC15 |
| BOC cc,disp | | 40XX | 41XX | 42XX | 43XX | 44XX | 45XX | 46XX | 47XX | 48XX | 49XX | 4AXX | 4BXX | 4CXX | 4DXX | 4EXX | 4FXX |
| | | AC0 | AC1 | AC2 | AC3 | | | | | | | | | | | | |
| LI r,disp | | 50XX | 51XX | 52XX | 53XX | | | | | | | | | | | | |
| sr / dr | | AC0 AC0 | AC1 AC0 | AC2 AC0 | AC3 AC0 | AC0 AC1 | AC1 AC1 | AC2 AC1 | AC3 AC1 | AC0 AC2 | AC1 AC2 | AC2 AC2 | AC3 AC2 | AC0 AC3 | AC1 AC3 | AC2 AC3 | AC3 AC3 |
| RAND sr,dr | | 5400 | 5440 | 5480 | 54C0 | 5500 | 5540 | 5580 | 55C0 | 5600 | 5640 | 5680 | 56C0 | 5700 | 5740 | 5780 | 57C0 |
| RXOR sr,dr | | 5800 | 5840 | 5880 | 58C0 | 5900 | 5940 | 5980 | 59C0 | 5A00 | 5A40 | 5A80 | 5AC0 | 5B00 | 5B40 | 5B80 | 5BC0 |
| RCPY sr,dr | | 5000 | 5C40 | 5C80 | 5CC0 | 5D00 | 5D40 | 5D80 | 5DC0 | 5E00 | 5E40 | 5E80 | 5EC0 | 5F00 | 5F40 | 5F80 | 5FC0 |
| | | AC0 | AC1 | AC2 | AC3 | | | | | | | | | | | | |
| PUSH r | | 6000 | 6100 | 6200 | 6300 | | | | | | | | | | | | |
| PULL r | | 6400 | 6500 | 6600 | 6700 | | | | | | | | | | | | |
| sr / dr | | AC0 AC0 | AC1 AC0 | AC2 AC0 | AC3 AC0 | AC0 AC1 | AC1 AC1 | AC2 AC1 | AC3 AC1 | AC0 AC2 | AC1 AC2 | AC2 AC2 | AC3 AC2 | AC0 AC3 | AC1 AC3 | AC2 AC3 | AC3 AC3 |
| RADD sr,dr | | 6800 | 6840 | 6880 | 68C0 | 6900 | 6940 | 6980 | 69C0 | 6A00 | 6A40 | 6A80 | 6AC0 | 6B00 | 6B40 | 6B80 | 6BC0 |
| RXCH sr,dr | | 6C00 | 6C40 | 6C80 | 6CC0 | 6D00 | 6D40 | 6D80 | 6DC0 | 6E00 | 6E40 | 6E80 | 6EC0 | 6F00 | 6F40 | 6F80 | 6FC0 |
| | | AC0 | AC1 | AC2 | AC3 | | | | | | | | | | | | |
| CAI r,disp | | 70XX | 71XX | 72XX | 73XX | | | | | | | | | | | | |
| sr / dr | | AC0 AC0 | AC1 AC0 | AC2 AC0 | AC3 AC0 | AC0 AC1 | AC1 AC1 | AC2 AC1 | AC3 AC1 | AC0 AC2 | AC1 AC2 | AC2 AC2 | AC3 AC2 | AC0 AC3 | AC1 AC3 | AC2 AC3 | AC3 AC3 |
| RADC sr,dr | | 7400 | 7440 | 7480 | 74C0 | 7500 | 7540 | 7580 | 75C0 | 7600 | 7640 | 7680 | 76C0 | 7700 | 7740 | 7780 | 77C0 |

Halt
Copy flags to register
Copy register to flags
Push flags onto stack
Pull stack into flags
Jump to subroutine; XX = ±127; push PC onto stack
Jump; XX = ±127
Exchange register and stack
Rotate register left
Rotate register right
Shift left
Shift right
} Bit 1 =1 include link bit; Bit 2 = 2 shift count; Bits 2–7 = N = shift count

Pulse or reset flag
Set flag

Branch on condition (PC relative) XX = ±127

Load immediate; load register with XX; XX = data
Bit 7 of XX extends to Bits 8–15 of register

"AND" register to register; result to register (dr)
Exclusive "OR" register to register; result to register (dr)
Copy register to register

Push register onto stack
Pull stack into stack

Add register to register; result to register (dr), overflow, and carry
Exchange register

Complement register and add XX; result to register
Bit 7 of XX is extended to Bits 8–15

Add register to register plus carry; result to register (dr); overflow and carry

# PACE INSTRUCTION SET

## ALPHANUMERIC SEQUENCE BY HEXADECIMAL

Read down then right.

| Mnemonic Assembler Code | | AC0 | AC1 | AC2 | AC3 | BASE PAGE (XX) | PC REL (XX + PC) | AC2 REL (XX + AC2) | AC3 REL (XX + AC3) |
|---|---|---|---|---|---|---|---|---|---|
| AISZ r, disp | | 78XX | 79XX | 7AXX | 7BXX | | | | |
| RTI disp | 7CXX | | | | | | | | |
| RTS disp | 80XX | | | | | | | | |
| DECA 0, disp(xr) | | | | | | 88XX | 89XX | 8AXX | 8BXX |
| ISZ disp(xr) | | | | | | 8CXX | 8DXX | 8EXX | 8FXX |
| SUBB 0, disp(xr) | | | | | | 90XX | 91XX | 92XX | 93XX |
| JSR @disp(xr) | | | | | | 94XX | 95XX | 96XX | 97XX |
| JMP @disp(xr) | | | | | | 98XX | 99XX | 9AXX | 9BXX |
| SKG 0, disp(xr) | | | | | | 9CXX | 9DXX | 9EXX | 9FXX |
| LD 0, @disp(xr) | | | | | | A0XX | A1XX | A2XX | A3XX |
| OR 0, disp(xr) | | | | | | A4XX | A5XX | A6XX | A7XX |
| AND 0, disp(xr) | | | | | | A8XX | A9XX | AAXX | ABXX |
| DSZ disp(xr) | | | | | | ACXX | ADXX | AEXX | AFXX |
| ST 0, @disp(xr) | | | | | | B0XX | B1XX | B2XX | B3XX |
| SKAZ 0, disp(xr) | | | | | | B8XX | B9XX | BAXX | BBXX |
| LSEX 0, disp(xr) | | | | | | BCXX | BDXX | BEXX | BFXX |
| LD r, disp(xr) | | X | | | | C0XX | C1XX | C2XX | C3XX |
| | | | X | | | C4XX | C5XX | C6XX | C7XX |
| | | | | X | | C8XX | C9XX | CAXX | CBXX |
| | | | | | X | CCXX | CDXX | CEXX | CFXX |
| ST r, disp(xr) | | X | | | | D0XX | D1XX | D2XX | D3XX |
| | | | X | | | D4XX | D5XX | D6XX | D7XX |
| | | | | X | | D8XX | D9XX | DAXX | DBXX |
| | | | | | X | DCXX | DDXX | DEXX | DFXX |
| ADD r, disp(xr) | | X | | | | E0XX | E1XX | E2XX | E3XX |
| | | | X | | | E4XX | E5XX | E6XX | E7XX |
| | | | | X | | E8XX | E9XX | EAXX | EBXX |
| | | | | | X | ECXX | EDXX | EEXX | EFXX |
| NE r, disp(xr) | | X | | | | F0XX | F1XX | F2XX | F3XX |
| | | | X | | | F4XX | F5XX | F6XX | F7XX |
| | | | | X | | F8XX | F9XX | FAXX | FBXX |
| | | | | | X | FCXX | FDXX | FEXX | FFXX |

Add XX to register; skip next instruction if result = zero; XX = ±127
Return from interrupt; add XX to top of stack and place result in PC; XX = ±127; set IEN flag
Return from subroutine; add XX to top of stack and place result in PC; XX = ±127
Decimal add register AC0 to contents of effective address; result to AC0, overflow and carry; address = (XX + register shown); XX = ±127
Increment contents of effective address by 1; skip next instruction if result = 0; result is in EA; use address mode shown; XX = ±127
Subtract contents of effective address from AC0; result to AC0; use address mode shown; XX = ±127
Jump to subroutine indirect; push PC onto stack; final address = to contents of location (XX + register shown); XX = ±127
Jump indirect; final address = to contents of location (XX + register shown); XX + ±127
Compare AC0 with contens of location (XX + register shown); XX = ±127; skip next instruction if AC0 > (EA)
Load indirect; load AC0 with contens of final address; address = contents of location (XX + register shown); XX = ±127
OR AC0 with contents of location (XX + register shown); XX = ±127; result to AC0
AND AC0 with contents of location (XX + register shown); XX = ±127; result to AC0
Decrement contents of effective address by 1; skip next instruction if result = 0; result is in EA; address = (XX + register shown); XX = ±127
Store indirect; store AC0 into final address; address = contents of location (XX + register shown); XX = ±127
AND AC0 with contents of location (XX + register shown); skip next instruction if result = 0; XX = ±127
Load AC0 with sign extended; Bit 7 of location (XX + register shown) is extended to AC0 8–15; Bits 0–7 are loaded to AC0 Bits 0–7; XX = ±127
Load AC0 with contents of location (XX + register shown); XX = ±127
Load AC1 with contents of location (XX + register shown); XX = ±127
Load AC2 with contents of location (XX + register shown); XX = ±127
Load AC3 with contents of location (XX + register shown); XX = ±127
Store AC0 to location (XX + register shown); XX = ±127
Store AC1 to location (XX + register shown); XX = ±127
Store AC2 to location (XX + register shown); XX = ±127
Store AC3 to location (XX + register shown); XX = ±127
Add AC0 to location (XX + register shown); XX = ±127; result to AC0
Add AC1 to location (XX + register shown); XX = ±127; result to AC1
Add AC2 to location (XX + register shown); XX = ±127; result to AC2
Add AC3 to location (XX + register shown); XX = ±127; result to AC3
Compare AC0 to location (XX + register shown); XX = ±127; if not equal skip next instruction
Compare AC1 to location (XX + register shown); XX = ±127; if not equal skip next instruction
Compare AC2 to location (XX + register shown); XX = ±127; if not equal skip next instruction
Compare AC3 to location (XX + register shown); XX = ±127; if not equal skip next instruction

# OTHER CLUBS AND PUBLICATIONS

A computer group with an excellent monthly publication is the Southern California Computer Society. For those of you considering microprocessors, an excellent article, "*The War of the Processors*," by Adam Osborne appears in the *May 1976* issue of their magazine. For further information contact:

Art Childs, Editor
Southern California Computer Society
P.O. Box 3123
Los Angeles, California 90051

An article by Gregory C. Jewell of Renton, Washington appears in the May issue of *BYTE*. "*Simply Your Homemade Assembler*" describes a simplified assembler language that can be used by PACE. For additional information on *BYTE* magazine contact:

BYTE
70 Main St.
Peterborough, N.H. 03458

# SC/MP HOMEBREW COMPUTER SYSTEM ADDITIONS

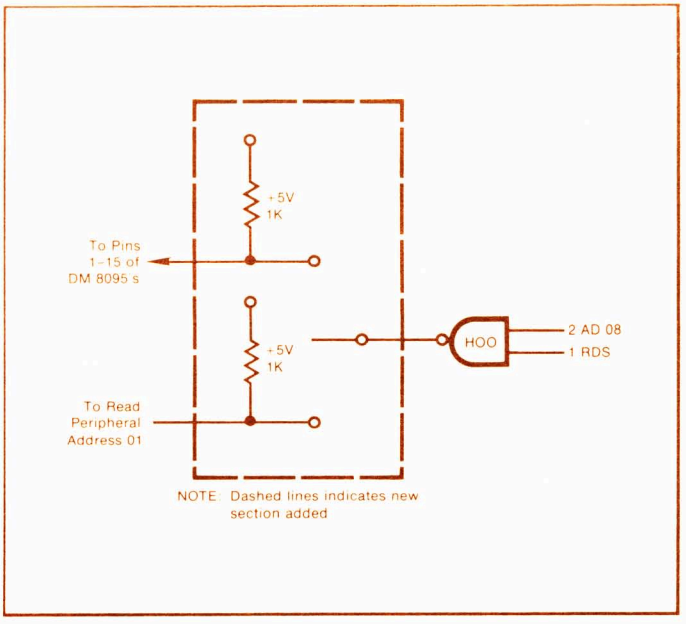
**by Dan Grove, μP Training, Santa Clara**

**1. Clock Components**
SC/MP Timing Element used was a 1000 pf capacitor across pins 37 and 38. Any crystal up to 1 Mhz will also work across the same pins. The only advantage of the more expensive crystal is software timing accuracy.

**2. Peripheral Device Additions and Addressing**
A peripheral device you intend only to write data into can share address 11 with the LEDs.

| Address Bits | | Device | Operation |
|---|---|---|---|
| 9 | 8 | | |
| 0 | 0 | PROM | READ |
| 0 | 1 | SWITCHES | READ |
| 1 | 0 | RAM | READ/WRITE |
| 1 | 1 | LEDS | WRITE |

Once a program is loaded into RAM, another switch can be used to disconnect the data switches so address 01 can be used as a peripheral device to read data as shown below:




# PROGRAM LIBRARY NEWS

If you have a listing of SL0012A, RAMDUMP, there is a change you should make. Otherwise, there will be a problem with dumping memory location 0. The fix requires a NOP instruction be inserted between lines 184 and 185 of the program listing. The corrected listing is printed on the following pages. We plan to be printing all library program listings in future issues of the newsletter.

One suggestion I would like to make to those of you who are copying the listing from the library. Verify that the source checksums agree, in case an undetected typing error has been made.

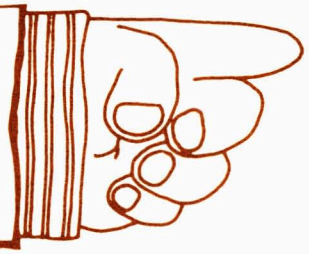
Also, we would like to thank everyone who has submitted programs or helped to improve the ones we have.
*Ed.*

```
                        .TITLE RAMDUMP
                        ;THIS PROGRAM PUNCHES OUT DATA FROM RAM
                        OR PROM
                        ;STANDARD RLM FORMAT IS USED
                        ;PROGRAM PROMPS ASKING FOR START AND
                        END ADDRESS
                        ;THE ADDRESS MUST CONTAIN 4 HEX DIGITS OR
                        ;PROGRAM REPROMPS
                        ;
                        ;
                        .TSECT
                        .GLOBL   BEG1
0000 812A   A BEG1:     LD       0,TCKSUM
0001 4000   A           PUSH     0
0002 2D7D   A           JSR      @MEGS
0003 002C   T           .WORD    STADDR
0004 294B   A           JSR      ASCBIN
0005 21FA   A           JMP      BEG1
0006 3781   A           RCPY     1,3          ;START→AC3
0007 2D78   A BEG2:     JSR      @MEGS
0008 0047   T           .WORD    ENADDR
0009 2946   A           JSR      ASCBIN
000A 21FC   A           JMP      BEG2
000B 295E   A           JSR      CMPARE
000C 21F3   A           JMP      BEG1
000D 4000   A           PUSH     0            ;SAVE WORD COUNT
000E 2D71   A           JSR      @MEGS
000F 0035   T           .WORD    TURNON
0010 2D15   A           JSR      @INTEST
0011 2101   A           JMP      .+2
0012 21FD   A           JMP      .-2
0013 4E1E   A           LI       2,30
0014 2D12   A           JSR      @NUL2
0015 296E   A           JSR      TITLE
0016 4400   A           PULL     0
0017 2D10   A           JSR      @CKSM
0018 5400   A           XCHRS    0
0019 3800   A           RADD     2,0
001A 5400   A           XCHRS    0
001B 2D0D   A           JSR      @DAREC
001C 21FA   A           JMP      .-5
001D 5400   A           XCHRS    0
001E 3800   A           RADD     2,0
001F 5400   A           XCHRS    0
0020 2D08   A           JSR      @DAREC
0021 2D08   A           JSR      @ENREC
0022 4E0A   A           LI       2,10
0023 2D03   A           JSR      @NUL2
0024 0000   A           HALT
0025 21DA   A           JMP      BEG1
0026 7EDF   A INTEST:   .WORD    07EDF
0027 009C   T NUL2:     .WORD    NULL2
0028 00A1   T CKSM:     .WORD    CKSUM
0029 00C0   T DAREC:    .WORD    DATARC
002A 00DC   T ENREC:    .WORD    ENDRC
002B E6E1   A TCKSUM:   .WORD    0E6E1
002C 5354   A STADDR:   .ASCII'START ADDR?
002D 4152   A
002E 5420   A
002F 4144   A
0030 4452   A
0031 3F20   A
0032 2020   A
0033 2020   A
0034 0000   A           .WORD    0
0035 5455   A TURNON:   ASCII'TURN PT PUNCHON AND HIT
                                             ANY KEY'
0036 524E   A
0037 2050   A
0038 5420   A
0039 5055   A
003A 4E43   A
003B 4820   A
003C 4F4E   A
003D 2041   A
003E 4E44   A
003F 2048   A
0040 4954   A
0041 2041   A
0042 4E59   A
0043 204B   A
0044 4559   A
0045 0D0A   A           .WORD    0D0A,0
0046 0000   A
0047 454E   A ENADDR:   .ASCII 'END ADDR?
0048 4420   A
0049 4144   A
004A 4452   A
004B 3F20   A
004C 2020   A
004D 2020   A
004E 2020   A
004F 0000   A           .WORD    0

                        ;SUBROUTINE GET 4 ASC DIGITS
                        ;CONVERT TO BINARY
                        ;REPROMP IF ILLEGAL INPUT
                        ;
                        ;
0050 4E04   A ASCBIN:   LI       2,4
0051 2D31   A LOOP:     JSR      @GECO
0052 6111   A           AND      0,MASK
0053 E112   A           SKG      0,HEX2F
0054 0200   A           RTS
0055 E111   A           SKG      0,HEX39
0056 2106   A           JMP      ARAB
0057 E110   A           SKG      0,HEX40
0058 0200   A           RTS
0059 E10F   A           SKG      0,HEX46
005A 2101   A           JMP      HIHEX
005B 0200   A           RTS
005C 4809   A HIHEX:    AISZ     0,9
005D 6107   A ARAB:     AND      0,MASK+1
005E 5D04   A           SHL      1,4
005F 3100   A           RADD     0,1
0060 4AFF   A           AISZ     2,-1
0061 21EF   A           JMP      LOOP
0062 3081   A           NOP
0063 0201   A           RTS      1
0064 007F   A MASK:     .WORD    07F,0F
0065 000F   A
0066 002F   A HEX2F:    .WORD    02F
0067 0039   A HEX39:    .WORD    039
0068 0040   A HEX40:    .WORD    040
0069 0046   A HEX46:    .WORD    046

                        .PAGE
                        ;SUBROUTINE COMPARE
                        ;COMPARES THE ADDRESSES FOR LEGALITY
                        ;FINDS THE ABSOLUTE DIFFERENCE BETWEEN THEM
                        ;AC3→START ADDR
                        ;AC1→END ADDR
                        ;AC0→NO. WORDS IN DUMP
                        ;
006A 3C81   A CMPARE:   RCPY     3,0
006B 1208   A           BOC      2,TT
006C 3481   A           RCPY     1,0
006D 1203   A           BOC      2,NOGOOD
006E 5001   A           CAI      0,1
006F 3C00   A           RADD     3,0
0070 1B01   A           BOC      11,.+2
0071 0200   A NOGOOD    RTS      0
```

## DOCUMENTATION UPDATES



This month we have updates to the PACE Technical Description, PACE Users Manual and the IMP-16C Applications Manual (see centerfold). Future changes will be printed in COMPUTE, if the change is small, or we will notify you of the change and tell you how to obtain it.

SUPPLEMENT 3 TO PUB. NO. 4200021C

IMP-16C Applications Manual

IMP-16C 200A/300A
Application Cards

May 1976

© National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, California 95051

## 1. INTRODUCTION

This supplement provides information to familiarize users with recent design improvements that have been made to the IMP-16C 200/300 microprocessor. The bulk of the information contained in the IMP-16C Application Manual still applies. The paragraphs that follow describe only those areas where changes have been made and apply only to the new version of the card — designated as IMP-16C 200A and IMP-16C 300A, and identifiable by the part number (5514736) on the printed wiring board.

## 2. POWER REQUIREMENTS

The +5-volt and -12-volt requirements indicated in section 1.3 remain unchanged, however, the use of a different memory device for on-card memory has eliminated the need for a -9-volt supply.

## 3. TIMING

The timing signals and timing diagrams described in chapter 4 remain unchanged. However, a Dual Voltage Controlled Oscillator (DM74S124) is now used to generate the master clock signal. This device replaces the triple line receiver and Schmitt Trigger circuit described in the first paragraph of section 4.1.

## 4. MEMORY

The MM1101 RAM devices previously used have been replaced by MM2101-1 devices, thus eliminating the need for -9-volt supply to the card. The amount of memory supplied with the card remains unchanged: 256 16-bit words of read/write memory, and sockets for 512 16-bit words of read-only memory. Memory selection and mapping remain as described in the IMP-16C Application Manual.

## 5. INITIALIZATION

The system initialization circuit described in section 4.7 has been replaced by the circuit being used in the IMP-16C 400/500 microprocessors. This circuit is fully described in SUPPLEMENT 2 (1.3.6) to the IMP-16C Application Manual.

## 6. DYNAMIC MEMORY INTERFACE

The Dynamic Memory Interface circuit described in section 7.2 is still provided on the card. The Refresh Request (RFREQ) and Cycle Initiate (CI) signals that were previously available via jumper pads are now available at card-edge connector pins 55 (RFREQ) and 83 (CI).

## 7. OPERATING PROCEDURES

The operating procedures described in section 8.7 still apply. One additional point should be observed: the Chip Select 3 (CS3) signal is reserved for possible future expansion of on-card memory. For proper operation of on-card RAM, no connection should be made to CS3 (pin 33).

## 8. OPTIONS

All of the options described in chapter 9 still apply as described.

## 9. LIST OF PIN CONNECTIONS AND SIGNALS

Appendix E (table E-1) lists the IMP-16C pin numbers and corresponding signal names. This table is still valid except for the following changes.

| Pin Number | Previous Designation | New Designation |
|---|---|---|
| 13, 14 | -9 Volts for R/W Memory | Not Used |
| 33 | Not Used | CS3 (DO NOT USE) |
| 55 | Not Used | RFREQ — Refresh Request |
| 83 | Not Used | CI — Cycle Initiate |

## 10. SCHEMATICS, PARTS LIST AND COMPONENT LAYOUT

The changes described in the preceding paragraphs have resulted in changes to the Schematic Diagram (figure 4-6), Parts List (table 4-2), and Component Layout (figure 4-8). Refer to engineering documentation supplied with the IMP-16C for up-to-date versions.

## 11. USE WITH IMP-16P

If the IMP-16C 200A/300A is to be used in an IMP-16P Microprocessor Development System, the following prewired jumper connections must be cut: W1, W2, W3, W6, and W7. With these connections cut, the card is pin compatible with and ready for use in the IMP-16P.

Technical Bulletin B76001 May 1976

Items Affected:

(1) PACE Technical Description (Pub. No. 4200078B)
(2) PACE Users Manual (Pub. No. 4200068X)

The following information is provided to supplement the design information currently available in the PACE Technical Description and in the PACE Users Manual. The information contained herein applies only to the IPC-16A/500D PACE microprocessor. Table 1 provides a cross-reference to assist in identifying those areas of existing PACE documentation where this supplemental information applies. If conflicts exist between the various documents, the information provided in this supplement takes precedence.

Table 1. PACE Documentation Cross-References

| Paragraph Number | Supplements Information In | | | |
|---|---|---|---|---|
| | PACE Technical Description (April 1976) | | PACE Users Manual | |
| | Page | Paragraph | Page | Paragraph |
| 1 | - - | - - | - - | - - |
| 2 | 2-30 | 2.5.4 | 3-25<br>4-15 | 3.12<br>4.12.1 |
| 3 | 2-34 | 2.5.6.1 | 4-10 | 4.8.1 |
| 4 | 2-32 | 2.5.5 | 2-16 | 2.9 |
| 5 | 2-33 | 2.5.5 | 4-15 | 4.11.2 |
| 6 | 2-33 | 2.5.5 | 4-18 | 4.12.4 |

1. HANDLING:

PACE utilizes high-impedance circuits. As with all devices of this type, high static charge environments should be avoided. Circuits should be kept in conductive carriers. In very dry environments, it may be desirable to ground personnel handling the package. Soldering irons and test equipment should be grounded.

2. HALT INSTRUCTION:

During programmed HALT, the NHALT output is true (low) (with a 7/8 duty cycle). The HALT instruction is terminated by the application of "CONTIN" pulse. The "CONTIN" input must go true (high) for a minimum of 16 clock cycles, and then low for four clock cycles for PACE operation to resume.

3. INITIALIZATION:

If the NINIT input is held true (low) while power and/or clocks are applied, the NADS and NHALT outputs may have an undefined state for 8 clock cycles after NINIT goes false (high). In order to initialize properly every time, NINIT should go true (low) after all the power supplies and clocks have stabilized. Thereafter, operation of the NINIT signal is as described in other documentation.

4. INTERRUPT DISABLE:

The use of PFLG or CRF instructions to disable the IEN flag allows one more instruction to be executed before the interrupts are disabled. If an interrupt should occur during execution of the PFLG or CRF instruction, the use of RTI would leave IEN true (one) after the execution of PFLG IEN. To prevent this situation the BOC instruction may be used to test PFLG or CRF instruction as follows:

```
PFLG IEN        ;TURN OFF IEN
BOC IEN, .-1    ;IS IEN FALSE?
                ;YES
```

5. STACK INTERRUPT:

If a stack interrupt occurs while there is a level-3 or level-4 interrupt present and enabled, the stack interrupt pointer will be accessed from location 0 instead of location 2. (This will not occur if the master interrupt enable, IEN, is false (zero) and is subsequently set true (one).) If the stack interrupt is used in conjunction with level-3 or -4 interrupts the contents of location 0 should therefore be the same as the contents of location 2. Since location 0 (zero) is also the initialize address, the opcode of the initialize instruction should be chosen to correspond to the stack interrupt pointer value. (For example, the unused field of a HALT instruction could be used to provide a stack interrupt pointer to any address in the first 1024 locations of memory.)

6. LEVEL ZERO INTERRUPT:

If a level-0 (zero) interrupt occurs within the 12-clock-cycle period (excluding extend cycles) following the recognition (indicated by CONTIN signal) of any other interrupt, the processor either will stall or execute the level-0 interrupt using the wrong pointer address. This problem may be avoided by only allowing the level-0 interrupt leading edge to be applied to the PACE chip during an NADS, provided no interrupt acknowledge has occurred since the last NADS.

The circuit shown in figure 1 is one means of accomplishing this. Note that the circuit has been designed to take care of proper "level-0" execution only. If one desires to "STALL" also, proper control gating will have to be added on to this circuit.

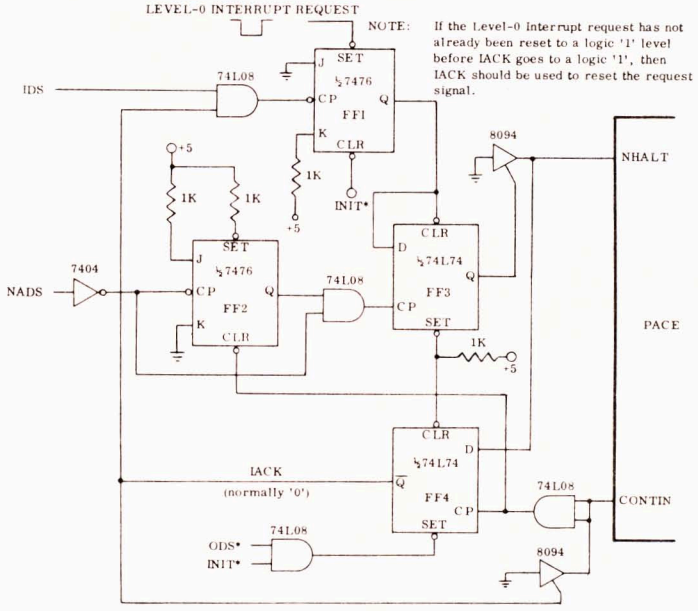



Figure 1. Circuit To Prevent Conflicts Between Level-0 and In-Process Interrupts

```
108 0072 5002    A            CAI     0,2
109 0073 0201    A            RTS     1
110 0074 3481    A TT:        RCPY    1,0
111 0075 1201    A            BOC     2,.+2
112 0076 2105    A            JMP     CMPUTE
113 0077 5001    A            CAI     0,1
114 0078 3C00    A            RADD    3,0
115 0079 1201    A            BOC     2,.+2
116 007A 2103    A            JMP     CMPUTE+2
117 007B 21F5    A            JMP     NOGOOD
118 007C 3C00    A CMPUTE:    RADD    3,0
119 007D 1201    A            BOC     2,.+2
120 007E 5002    A            CAI     0,2
121 007F 0201    A            RTS     1
122 0080 7EC3    A MEGS:      .WORD   07EC3
123 0081 7E59    A PUTC:      .WORD   07E59
124 0082 7ED3    A PUT2C:     .WORD   07ED3
125 0083 7E73    A GECO:      .WORD   07E73
126
                              .PAGE
127                           ;SUBROUTINE TITLE RECORD
128                           ;PUNCHES TITLE IN RLM FORMAT
129                           ;
130 0084 8112    A TITLE:     LD      0,STEX
131 0085 2DFC    A            JSR     @PUT2C
132 0086 4C05    A            LI      0,5
133 0087 2DF9    A            JSR     @PUTC
134 0088 81A2    A            LD      0,TCKSUM
135 0089 2DF8    A            JSR     @PUT2C
136 008A 4E02    A            LI      2,2
137 008B 2910    A            JSR     NULL2
138 008C 810B    A            LD      0,NAME
139 008D 2DF4    A            JSR     @PUT2C
140 008E 810A    A            LD      0,NAME+1
141 008F 2DF2    A            JSR     @PUT2C
142 0090 8109    A            LD      0,NAME+2
143 0091 2DF0    A            JSR     @PUT2C
144 0092 8108    A            LD      0,CRLF
145 0093 2DEE    A            JSR     @PUT2C
146 0094 4E04    A            LI      2,4
147 0095 2906    A            JSR     NULL2
148 0096 0200    A            RTS
149 0097 0200    A STEX:      .WORD   0200
150 0098 4D41    A NAME:      .WORD   'MA'
151 0099 494E    A            .WORD   'IN'
152 009A 5052    A            .WORD   'PR'
153 009B 0D0A    A CRLF:      .WORD   0D0A
154                           ;SUBROUTINE WRITE NULLS
155 009C 4C00    A NULL2:     LI      0,0
156 009D 2DE4    A            JSR     @PUT2C
157 009E 4AFF    A            AISZ    2,-1
158 009F 21FD    A            JMP     .-2
159 00A0 0200    A            RTS
160
                              .PAGE
161                           ;SUBROUTINE CHECK SUM
162                           ;COMPUTES CKSM FOR EACH RECORD
163                           ;AC2→CHECKSUM
164                           ;AC3→LOAD ADDRESS
165                           ;AC0→
166                           ;AC1→RECORD LENGTH
167                           ;
168 00A1 4000    A CKSUM:     PUSH    0
169 00A2 3E81    A            RCPY    3,2
170 00A3 4D0C    A            LI      1,12
171 00A4 8300    A OVER:      LD      0,(3)
172 00A5 3200    A            RADD    0,2
173 00A6 4B01    A            AISZ    3,1
174 00A7 3081    A            NOP
175 00A8 4400    A            PULL    0
176 00A9 48FF    A            AISZ    0,-1
177 00AA 2101    A            JMP     .+2
178 00AB 2108    A            JMP     OUT2
179 00AC 49FF    A            AISZ    1,-1
180 00AD 2101    A            JMP     .+2
181 00AE 2102    A            JMP     OUT1
182 00AF 4000    A            PUSH    0
183 00B0 21F3    A            JMP     OVER
184 00B1 4D10    A OUT1:      LI      1,16
185 00B2 4FB4    A            AISZ    3,-12
186 00B3 0200    A            RTS
187 00B4 4C11    A OUT2:      LI      0,17
188 00B5 5101    A            CAI     1,1
189 00B6 3400    A            RADD    1,0
190 00B7 4000    A            PUSH    0
191 00B8 4C0C    A            LI      0,12
192 00B9 3400    A            RADD    1,0
193 00BA 5001    A            CAI     0,1
194 00BB 3300    A            RADD    0,3
195 00BC 4BFF    A            AISZ    3,-1
196 00BD 3081    A            NOP
197 00BE 4500    A            PULL    1
198 00BF 0205    A            RTS     5
199
                              .PAGE
200                           ;SUBROUTINE PUNCH DATA RECORDS
201                           ;AC2→CHECKSUM
202                           ;AC3→LOAD ADDRESS
203                           ;AC1→RECORD LENGTH
204                           ;
205 00C0 4000    A DATARC:    PUSH    0
206 00C1 8119    A            LD      0,STEXD
207 00C2 2DBF    A            JSR     @PUT2C
208 00C3 3481    A            RCPY    1,0
209 00C4 2DBC    A            JSR     @PUTC
210 00C5 3881    A            RCPY    2,0
211 00C6 2DBB    A            JSR     @PUT2C
212 00C7 4E01    A            LI      2,1
213 00C8 29D3    A            JSR     NULL2
214 00C9 3C81    A            RCPY    3,0
215 00CA 2DB7    A            JSR     @PUT2C
216 00CB 4E02    A            LI      2,2
217 00CC 29CF    A            JSR     NULL2
218 00CD 49FC    A            AISZ    1,-4
219 00CE 3081    A            NOP
220 00CF 8300    A            LD      0,(3)
221 00D0 2DB1    A            JSR     @PUT2C
222 00D1 4B01    A            AISZ    3,1
223 00D2 3081    A            NOP
224 00D3 49FF    A            AISZ    1,-1
225 00D4 21FA    A            JMP     .-5
226 00D5 81C5    A            LD      0,CRLF
227 00D6 2DAB    A            JSR     @PUT2C
228 00D7 4E04    A            LI      2,4
229 00D8 29C3    A            JSR     NULL2
230 00D9 4400    A            PULL    0
231 00DA 0200    A            RTS
232 00DB 0280    A STEXD:     .WORD   0280
233                           ;SUBROUTINE PUNCH END RECORD
234                           ;
235 00DC 810E    A ENDRC:     LD      0,STEXEN
236 00DD 2DA4    A            JSR     @PUT2C
237 00DE 4C04    A            LI      0,4
238 00DF 2DA1    A            JSR     @PUTC
239 00E0 4400    A            PULL    0
240 00E1 5400    A            XCHRS   0
241 00E2 2D9F    A            JSR     @PUT2C
242 00E3 4000    A            PUSH    0
243 00E4 4E03    A            LI      2,3
244 00E5 29B6    A            JSR     NULL2
245 00E6 4400    A            PULL    0
246 00E7 2D9A    A            JSR     @PUT2C
247 00E8 81B2    A            LD      0,CRLF
248 00E9 2D98    A            JSR     @PUT2C
249 00EA 0200    A            RTS
250 00EB 02C0    A STEXEN:    .WORD   02C0
251      0000                 .END

ARAB    005D T ASCBIN   0050 T BEG1     0000 GT
BEG2    0007 T CKSM     0028 T CKSUM    00A1  T
CMPARE  006A T CMPUTE   007C T CRLF     009B  T
DAREC   0029 T DATARC   00C0 T ENADDR   0047  T
ENDRC   00DC T ENREC    002A T GECO     0083  T
HEX2F   0066 T HEX39    0067 T HEX40    0068  T
HEX46   0069 T HIHEX    005C T INTEST   0026  T
LOOP    0051 T MASK     0064 T MEGS     0080  T
NAME    0098 T NOGOOD   0071 T NUL2     0027  T
NULL2   009C T OUT1     00B1 T OUT2     00B4  T
OVER    00A4 T PUT2C    0082 T PUTC     0081  T
STADDR  002C T STEX     0097 T STEXD    00DB  T
STEXEN  00EB T TCKSUM   002B T TITLE    0084  T
TT      0074 T TURNON   0035 T
NO ERROR LINES
END PASS 2
SOURCE CHECKSUM=13A0
OBJECT CHECKSUM=AFF7
```

# A DATA CONCENTRATOR USING PACE

**by Barney Hordos**
**National Semiconductor**
**Santa Clara, California**

INTRODUCTION:

A data concentrator is a device which takes data from several different slow speed lines and re-transmits them along a single higher speed line. A microprocessor is an ideal device to perform this function because of its versatility. This application note describes the use of National's PACE microprocessor as a data concentrator.

In this application, PACE is used to collect information from several teletype or CRT terminals and re-transmits this information to a large computer over a single telephone line. PACE will also receive information from the large scale computer and direct data to the desired terminal. Figure 1 is a block diagram of the system.

Since teletypes normally operate at slow transmission rates (10–30 cps) and a modem can operate at high speeds (4800 baud), there is a waste of modern capabilities if one modem per teletype is used. By connecting several terminals to one modem, a more efficient bandwidth is utilized. This results in higher throughput and lower communications cost.

For this operation, the following equipment is required:

1. A PACE CPU, associated interface chips and memory.
2. A standard full duplex telephone modem.
3. A processor/modem interface.
4. A terminal controller for each terminal.
5. A simple priority controller for the terminal controller.

THE PACE PROCESSOR:

PACE (Processing And Control Element) is a single-chip 16 bit microprocessor packaged in a 40-pin dual in-line package. Around PACE are interfacing chips to buffer and demultiplex the outputs of PACE.

INTERFACE AND MEMORY:

The STE is the clock generator to form the MOS clocks necessary for PACE and also a TTL compatible clock for the rest of the system.

The data bus is buffered and interfaced using two BTE chips. The BTE (Bidirectional Transceiver Element) is an 8-bit interface between PACE and a TTL bus. One BTE is also used to interface the flags and control lines to TTL levels.

The address is latched using an ALE (Address Latch Element) which is a 16-bit storage latch.

The Memory is configured as 16-bits wide with part of the memory in ROM or PROM and part as RAM. For further information see reference 2.

THE FULL DUPLEX MODEM

The modem can be any one of the many available on the market today. It must be full duplex so PACE can handle both modem inputs and outputs on a simultaneous basis. In addition, the modem should be capable of operating at 4800 baud.

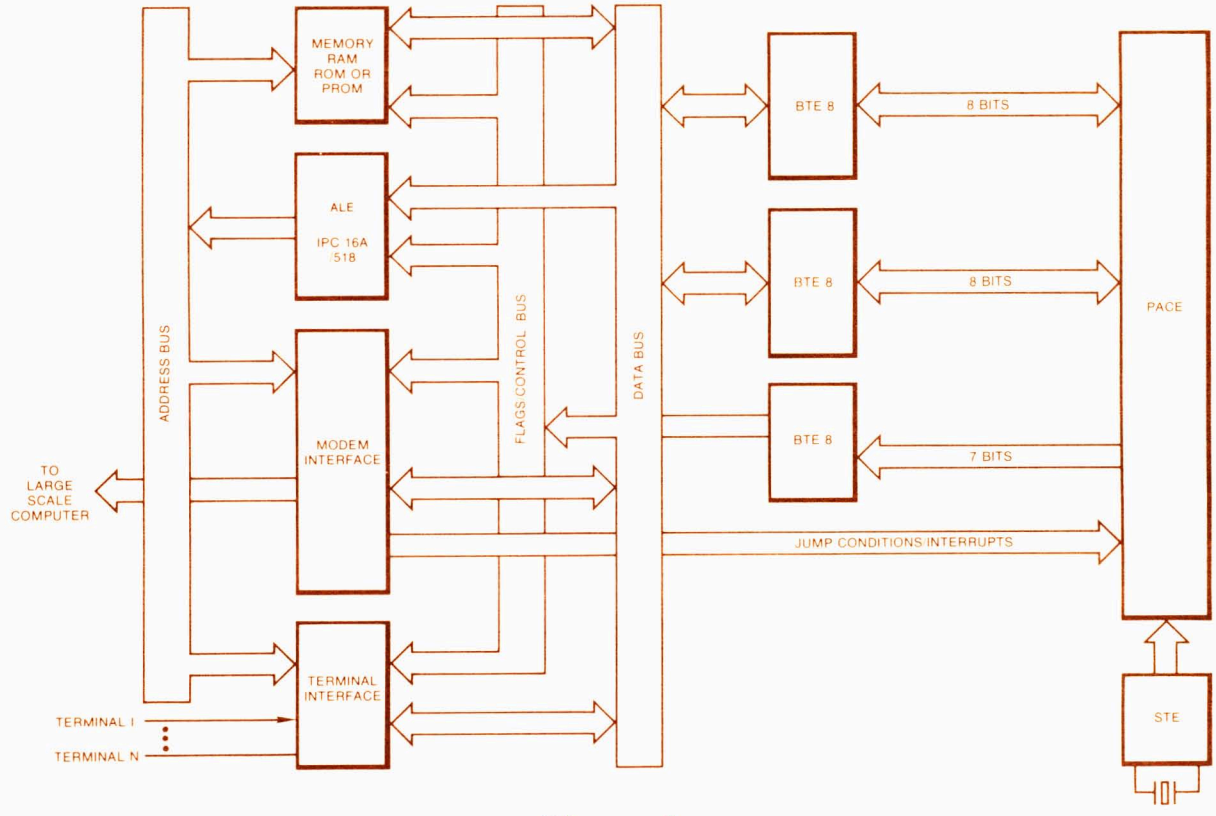



**Figure 1**

## THE PROCESSOR/MODEM INTERFACE

The interface is completely serial and as such can take advantage of the user jump condition and control flags of PACE. Figure 2 shows all the circuitry needed for transmission up to 4800 baud. The jump condition can be tested under program control using the branch on condition instruction. The input sampling rate is controlled by a software delay routine which generates the proper bit rate.

## THE TERMINAL CONTROLLER INTERFACE

There is one terminal controller for each terminal. Using National's MM5303 UAR/T and MM5307 Programmable Baud Rate Generator and associated buffering a controller can be built, see Figure 3. The controller will have an input buffer, an output buffer, a status register and control for timing. To output a character to a specific terminal, first the status must be checked. If the controller is busy (ie, already outputting a character), the busy line will be true. When the busy line goes false the next character can be sent. When the controller has an input character it will generate an interrupt to the priority encoder. The priority encoder generates the address of the highest priority interrupting terminal.

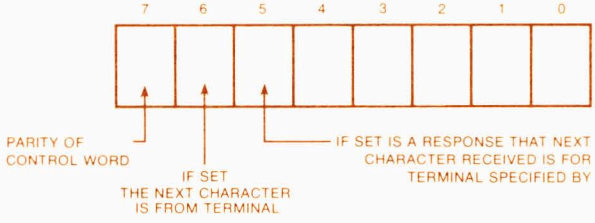
## THE PRIORITY ENCODER CONTROLLER

The interrupt from each terminal is connected to the priority encoder controller. The controller then generates the address of the highest priority interrupting device, which is put on the data bus as commanded by PACE. If no device is interrupting, an address of zero will be put on the bus. After it is determined which device is interrupting, PACE then receives the character from the terminal, which resets its interrupt, allowing other devices, if interrupting, to now be addressed. Figure 4 is a simplified block diagram of the Priority encoder controller.

## PACE CONTROL PROGRAM

The control program handles all of the communication with the user terminals and also with the large scale computer. After all of the terminals are initialized PACE communicates with the large computer and logs on to its system, this is the initialize portion of the program. The main program consists of checking if any terminal has a character. If the terminal has a character, the data is read and the output control character is formed and transmitted to the large computer, along with the data. A response is received from the large computer that signifies that the character pair was or was not received correctly and if the large computer has some data for PACE. When PACE receives a character pair from the large computer, PACE sends a response indicating the pair was received and if PACE had data for the large computer. This program would be contained in either ROM or PROM. For each data character transmitted to the modem, there will also be one control character sent; similarly for each data character received there is a control character associated with it.

THE OUTPUT CONTROL CHARACTER will have the following format:




Bit 5 of the output control character, if set, signifies that PACE is ready to receive a character. If it is reset PACE cannot receive character and the large scale

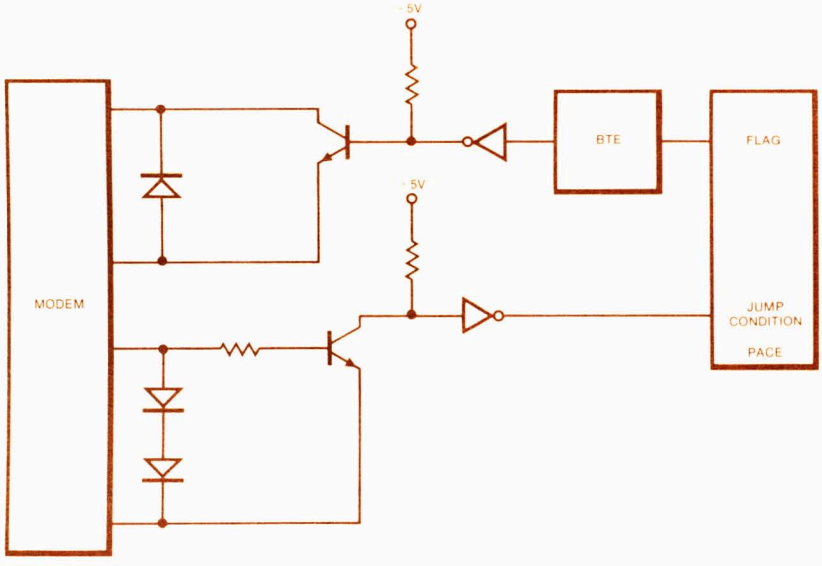
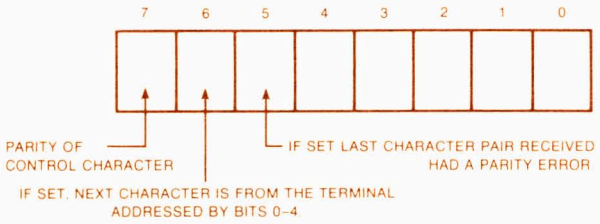



**Figure 2**

computer must refile that character. Bit 6, if set, signifies that the next character transmitted by PACE will be from the terminal addressed by bits 0–4. If both bits 5 and 6 are reset, this implies the last character transmitted by the large scale computer to PACE was received with a parity error and the character should be re-transmitted.

THE INPUT CONTROL CHARACTER has the following format:


7 6 5 4 3 2 1 0
PARITY OF CONTROL CHARACTER
IF SET, NEXT CHARACTER IS FROM THE TERMINAL ADDRESSED BY BITS 0–4
IF SET LAST CHARACTER PAIR RECEIVED HAD A PARITY ERROR

Each time the large scale computer receives an output control character from the modem it will reply with an input control character, if bit 6 is set, transmit one data word. If bit 5 is set, this implies that the last word sent from PACE had a parity error and should be resent. Figure 5 is a flow chart which shows all of the basic control functions performed by the control program stored in ROM or PROM.

How many terminals can PACE control using this manner?

Operating the modem at 4800 baud will allow PACE to transmit 480 characters per second. Since each data word must have a control word associated with it, this now reduces the character rate to 240 characters per second. If all the terminals operate at 10 cps, the maximum number of terminals would be 24.8. There are other factors to be considered. The worst case is if all terminals generate an interrupt simultaneously. All terminals must be serviced in sequence and no data lost or no terminal must be locked out. Since PACE could be at the beginning of input routine from the modem when all terminals request service, PACE must complete the receive routine before it can begin to service the terminals. This delay corresponds to one character pair time, or one terminal service time. Without considering

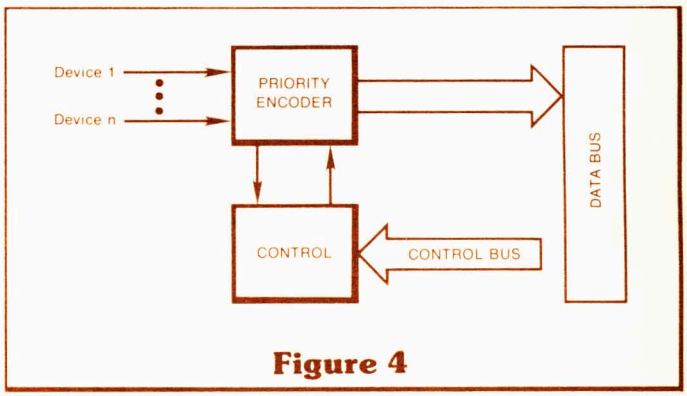



**Figure 4**

any other delays the total number of terminals possible would be 23.8. Since the compute time between characters is minimal compared to the character transmit time, it can be neglected. However, consider the case of handling parity. Many applications, such as text editors, don't require parity checking since the editor inherently performs validity checking of its own, so no parity checking is required. If parity is required, the maximum number of terminals is directly proportional to the worst acceptable error rate. Since each parity error requires a re-transmission, each parity error will correspond to one terminal service time. Therefore, a 50% error rate will limit the number of terminals to 11; a 25% error rate would limit the number of terminals to 15 and so on.

CONCLUSIONS:

There are many applications where this data concentrator scheme could be used. The total number of terminals is dependent upon the type and speed of terminals as well as the speed of the modem transmission rate.

The main advantage of this scheme is that it eliminates the need to have one modem for each terminal. This will reduce the number of computer connections and also reduce telephone rates.

REFERENCES

1. PACE USERS MANUAL (#4200068)
2. PACE INTERFACE DEVICES
3. THE IMP-16 IN COMMUNICATION APPLICATION (AN134)
4. THE IMP-8C AS A DATA CONCENTRATOR (AN113)

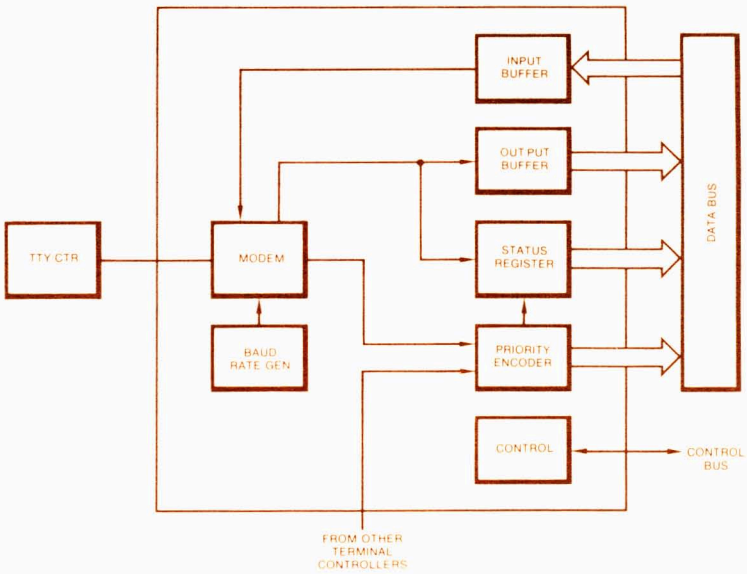



**Figure 3**

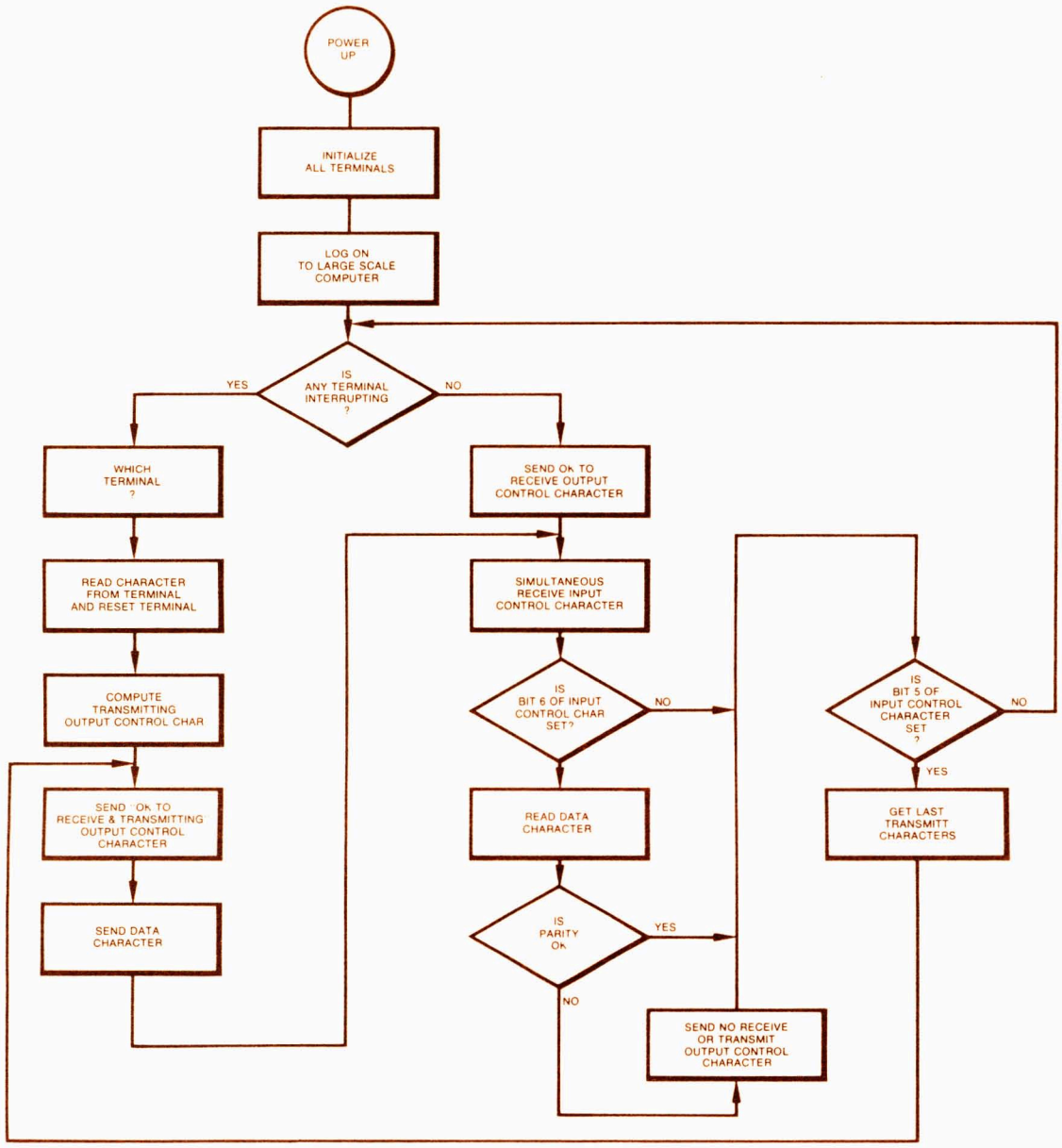

POWER UP
INITIALIZE ALL TERMINALS
LOG ON TO LARGE SCALE COMPUTER
IS ANY TERMINAL INTERRUPTING ?
YES
NO
WHICH TERMINAL ?
READ CHARACTER FROM TERMINAL AND RESET TERMINAL
COMPUTE TRANSMITTING OUTPUT CONTROL CHAR
SEND OK TO RECEIVE & TRANSMITTING OUTPUT CONTROL CHARACTER
SEND DATA CHARACTER
SEND OK TO RECEIVE OUTPUT CONTROL CHARACTER
SIMULTANEOUS RECEIVE INPUT CONTROL CHARACTER
IS BIT 6 OF INPUT CONTROL CHAR SET?
NO
READ DATA CHARACTER
IS PARITY OK
YES
NO
SEND NO RECEIVE OR TRANSMIT OUTPUT CONTROL CHARACTER
IS BIT 5 OF INPUT CONTROL CHARACTER SET ?
NO
YES
GET LAST TRANSMITT CHARACTERS

**Figure 5**

# AN IMP-16 MICROCOMPUTER SYSTEM PART 2

**by Hal Chamberlin**

**Reprinted with permission from The Computer Hobbyist Number 9/February, 1976**

Things are changing so rapidly that the first paragraph of these installments will have to be devoted to news items. Poly-Paks no longer has IMP-16 sets. We don't know if IEU still has them or not. However all surplus IMP-16 chip sets come through Godbout so perhaps some letters will persuade him to sell them directly. Of course all National distributors have some; TCH has gotten them this way for $160. The real problem is that they hit the surplus market too *early*. We got some more data on the "power math" CROM. Basically it provides instructions for operating on 32 bit binary fractions (mantissas) such as 32X32 add, subtract, multiply, divide, and normalize. The user need only code exponent handling and the result is a floating point package with 32 bit mantissas (10 decimal digits) and 16 bit exponents (10**10000 anybody?) with a 100 $\mu$s add time and 600 $\mu$s multiply time. The bad news is that "power math" and the extended CROM share some op-codes so they cannot normally be used together. There is a way to enable one or the other using a status flag however (status flags can be saved during interrupt). Implementation of the scheme requires the use of a 74LS260 in place of the 74LS54. PC layout of the CPU board is planned but some readers couldn't wait and have already started to wire-wrap CPU boards. At least 3 TCH staff members will be building IMP systems and at least one of them will have a floppy disk so software support will not be lacking toward summer. Quick note: do not buy plain 2107 4K RAM's for this system! They have a different pinout, are very slow, and in a word, totally obsolete. TMS4030, TMS4060, 2107A, and 2107B are all fine as well as most gradeouts. The author has a limited supply of TMS4030-ZA0248 4K RAMS tested for operation in this system

for $7.50. An error was made in the parts list for the memory board. Rather than three 7404's, it should be two 7404's and a 7440.

Now with the news out of the way, let us take a top-down approach to describing the PUNIBUS controller. The bus controller runs continuously, non-stop, from power-up to power-down crunching out 1.43 million cycles per second or one cycle every 700NS. All memories in the system likewise operate at this cycle rate. Each cycle is awarded on the basis of priority to one of 7 possible requesters. The highest priority requester is the CPU. Below the CPU are 5 direct memory access (DMA) devices. The lowest priority requester is the memory refresher which is always requesting bus cycles. Thus if the system is idle, that is, CPU halted and no DMA activity, all of the bus cycles are being awarded to the memory refresher. During operation, cycles that are unclaimed by the CPU or DMA are also awarded to the refresher. The PUNIBUS controller always generates the timing signals necessary for data transfer regardless of which requester controls the particular cycle. Thus DMA devices in the system don't have to generate any timing of their own, instead they just sit and respond to control signals issued by the PUNIBUS controller.

Any device interfaced to the bus that is not a possible DMA requester is expected to behave as if it was a memory. At the beginning of every bus cycle a 16 bit address is established. This address specifies either an actual memory location or a peripheral device register. There are only two types of bus cycles; a read cycle and a write cycle. During a read cycle, data is read from a memory or peripheral register into the CPU or DMA device. During a write cycle, data is written from the CPU or DMA device into a memory or peripheral register. The CPU or DMA device awarded the cycle determines whether a read or a write cycle is to be performed. The memory refresher, of course, always does read cycles. Undefined operations such as addressing non-existent memory or writing into a read-only peripheral register are not harmful and function as NO-OP bus cycles.

Figure 1 shows the timing relationships of the PUNIBUS. Although actual times in nanoseconds are given, it is important to note that correctly designed interfaces to the bus will work properly even with considerable variation in the timing details as long as the basic relationships are retained. This allows flexibility to change the details to accomodate other CPU's such as a bipolar IMP or a down-spec chip set without obsoleting memory and peripheral designs.

As can be seen, a bus cycle starts with the signal BUS ADDRESS ENABLE (BAE) going high and terminates when it goes high again for the next cycle. Actually though, some preparation takes place toward the end of the previous cycle. An internal "priority strobe" is generated which causes the BUS REQUEST (BR) lines including CPU and refresh request to be examined to determine who will get the next cycle. The determination is made and the three bit grant code of the winning requestor is placed on the BUS GRANT (BG) lines immediately before the cycle commences with BAE going high. At this time the one requester whose code is on the BG lines is expected to gate a 16 bit address onto the BUS DATA (BD) lines as long as BAE is high. Any BD lines not specifically driven will assume a ONE level because of pullup resistors. If a write cycle is to be executed, the BUS WRITE REQUEST (BWR) line should be pulled down during BAE time, otherwise a read cycle will be automatically assumed. This address phase of the cycle is identical for both read and write operation.

After the address phase we have the data transfer phase which is different for read and write cycles. In the case of a read cycle, the bus controller generates two signals, BUS DATA OUT ENABLE (BDOE) and BUS DATA OUT STROBE (BDOS) which control the data transfer from memory or peripheral register to CPU or DMA device. BDOE first goes high to cause the addressed memory or peripheral to gate its data onto the BD lines. BDOS is bracketed by BDOE and can be used to strobe data from the bus into the CPU or DMA device's data register on its trailing edge. The timing of this pair of pulses is chosen to allow memories sufficient access time and to allow the IMP-16 chip set to grab the data directly from the bus with no intervening latches.

During the transfer phase of a write cycle, BDOE and BDOS remain inactive while BUS DATA IN ENABLE (BDIE) and BUS WRITE ENABLE (BWE) control the data

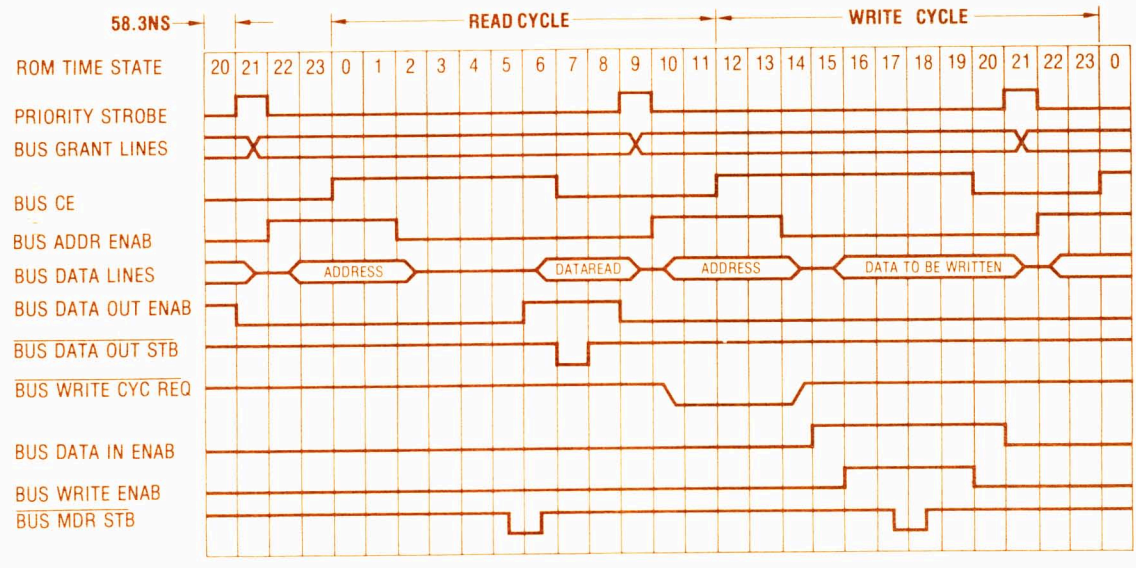

58.3NS
READ CYCLE
WRITE CYCLE
ROM TIME STATE
20 21 22 23 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0
PRIORITY STROBE
BUS GRANT LINES
BUS CE
BUS ADDR ENAB
BUS DATA LINES
BUS DATA OUT ENAB
BUS DATA OUT STB
BUS WRITE CYC REQ
BUS DATA IN ENAB
BUS WRITE ENAB
BUS MDR STB
ADDRESS
DATAREAD
ADDRESS
DATA TO BE WRITTEN

**Figure 1. *PUNIBUS Timing Relationships***

transfer from CPU or DMA to memory or peripheral. BDIE becomes active first causing the CPU or DMA device to gate the data to the written onto the BD lines. BWE which is bracketed by BDIE then becomes active causing the memory or peripheral to accept and store data from the bus. The timing shown for these signals was chosen to be compatable with the 4K RAM's used in this system.

The responsibilities of a memory board or peripheral interface are quite simple. During the address phase of each cycle, pertinent information about the 16 bit address on the BD lines must be latched on each interface board. Generally, a memory interface using 4K RAM's need only latch a single bit since the RAM chips have built-in latches for address and chip select. The single bit needing a TTL latch simply indicates whether the board was addressed or not. Likewise, a peripheral register can decode its address directly from the BD lines and use a flip-flop to remember if it was addressed. In either case, the leading edge of BUS CE is used for address strobing since it always occurs when the address is valid. Once a memory or peripheral has latched the fact that it was addressed, it either sends its data out if it sees BDOE or accepts new data in if it sees BWE. Thus memories and peripheral registers are passive, merely responding to bus signals as they occur.

Four "convenience" signals are provided on the bus. One which has already been mentioned in BUS CE. Memory boards using 22 pin 4K RAM's can simply amplify this signal to NMOS levels and apply it to the Chip Enable clock input of the RAM chips. Its function within the RAM is to start up the memory cycle and also strobe the on-chip address and chip select latches. Another signal provided specifically for memory boards is $\overline{\text{BUS MDR STOBE}}$. Its purpose is to strobe the data out latches on the memory board when data out from the 4K RAM's is valid. Some unfortunate timing constraints on both TMS4030 and 2107 type RAM's require latches to hold the data after it disappears from the RAM outputs. Although BDOE could have been turned on earlier with the leading edge strobing the latches, excessive noise generation would have resulted. $\overline{\text{BUS I/O ADDR}}$ is a signal that goes low whenever the binary value on the data lines is between FF00 and FF7F hexadecimal. This range of addresses is normally assigned to peripheral devices. Use of this signal in decoding I/O addresses can save a 9-input AND gate equivalent on each interface card. BUS CLOCK is provided as a convenient high frequency clock with a .005% accuracy. Its frequency is such that when put through a 16 bit divider, the resulting frequency is middle C, 261.625 Hz. Additionally, 12 cycles of this clock make up one bus cycle whose length is actually 699.88NS.

Two signals are involved with power-on reset and console reset. The $\overline{\text{POWER OK}}$ bus line should be pulled low by an external circuit associated with the power supply when all supply voltages are present and stabilized. This circuit should also be connected to the console reset push button so a power-on sequence can be simulated without losing memory contents. A simple delay circuit is shown in figure 2 which functions quite well. Alternatively, a true power monitor can be built using zener diodes to sense when the supply voltages are actually present. $\overline{\text{BUS RESET}}$ is generated in response to $\overline{\text{POWER OK}}$ by the CPU board. It resets the CPU and should reset all peripheral interfaces to a safe, idle

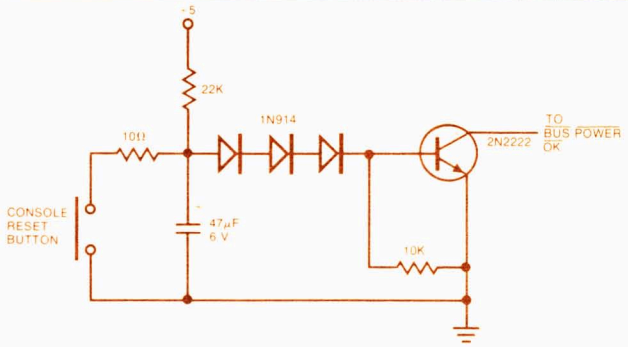



**Figure 2. *Simple Power OK Circuit***

condition when it goes low. It has no effect on the bus controller or memory refresher however.

The interrupt system uses the very simple software polling technique described elsewhere in this issue. The $\overline{\text{BUS INT REQ (BIR)}}$ line is a wire-or line with pullup resistor which is pulled low by any device that wants to request an interrupt. The CPU responds, provided its master interrupt enable is on, by calling a subroutine at 0001 and simultaneously turning master interrupt enable off. After saving status, the program can look at the status register of each possible interrupting device to determine who is requesting. This search can be as fast as 9.8$\mu$S per device with proper use of the SKAZ (Skip if And is Zero, ANDs addressed memory location with a register and skips the next instruction if the result is zero) instruction. The device service routine then turns off the interrupt request for that particular device and turns master interrupt enable back on. Priority in the case of simultaneous interrupts is determined by the order of scanning. Nested interrupts can also be programmed. Thus the interrupt system essentially works like that on a PDP-8. The usual interrupting device interface also has an interrupt enable for each device making non-interrupt I/O programming possible if desired. More details on I/O interfacing and interrupts will be given in part 4.

Figures 3 and 4 show the timing generator and bus controller. Since this circuitry is on the CPU board, some CPU circuitry has encroached which will be described in part 3. See TCH #2 if any of the logic gate symbols are confusing. You will note that inputs always enter from the left of a drawing and outputs leave at the right. All signals going offpage are given a name and should mate with similarly named signals on the other pages. If an offpage signal has a number on it, it goes to the CPU board edge connector. If the number is 46 or less, it is a bus signal and is available at the same pin number of any board in the system. Some signals shown in figure 3 and 4 will not be mated until part 3.

The heartbeat of the system is the 17.145893 mHz oscillator in figure 3. Its output drives a hex latch and is buffered to drive the $\overline{\text{BUS CLOCK}}$ line. The latch and two 32-word by 8-bit bipolar PROM's make up the bulk of the timing generator. As can be seen, 6 of the 16 PROM outputs go to the 74S174 hex latch and 5 of these are fed back to the PROM address inputs. The results is that every cycle of the 17 mHz clock causes the PROM-latch combination to take one step in a programmed sequence. Using the PROM pattern in figure 5, this sequence is 24 steps long and takes 1.4$\mu$s to step through thus matching the minimum IMP-16 microcycle time. In order to avoid

| ROM ADDRESS | TIME STATE | CPU-1 RAW CE | RAW ADDR ENAB | RAW MDR STROBE | NEXT ROM ADDR 16 | NEXT ROM ADDR 8 | NEXT ROM ADDR 4 | NEXT ROM ADDR 2 | NEXT ROM ADDR 1 | CPU-2 PRIORITY STROBE | CPU DATA IN ENAB | CPU LOAD REG | CPU FLAG ADDR STB | RAW WRITE ENAB | RAW DATA IN ENAB | RAW DATA OUT STB | RAW DATA OUT ENAB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 16 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 15 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 2 | 23 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | — | X | X | X | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 4 | — | X | X | X | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 5 | 14 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 7 | 7 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 8 | 17 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 9 | 10 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 10 | — | X | X | X | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 11 | 9 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 12 | 18 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 13 | — | X | X | X | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 14 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 15 | 8 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 16 | 21 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 17 | — | X | X | X | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 18 | 22 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 19 | 5 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 20 | 20 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 21 | 13 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 22 | — | X | X | X | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 23 | 6 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 24 | — | X | X | X | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 25 | 11 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 26 | 3 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 27 | 4 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 28 | 19 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 29 | 12 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 30 | 2 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 31 | — | X | X | X | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |

**Figure 3. *Timing ROMS in Address Sequence***

glitches at the PROM outputs when the address changes, the sequence of addresses has been chosen such that only one address line changes at a time. Figure 6 shows the PROM pattern in time sequence rather than address sequence. The 8 addresses not normally used all point to time state zero to avoid a possible lockup condition. The sequence of addresses was also chosen so that a decoder could be used to generate the 4-phase non-overlapping clock needed by the IMP-16 chips from 3 of the address bits.

The remaining 11 PROM bits are the various system timing signals. Those prefixed RAW require additional gating before being used; the others are ready to go. BUS MDR STROBE goes through the latch to effect an additional 30ns delay. The purpose of the flip-flop, connected to the 4-phase decoder is to insure that the CPU starts up on phase 1 after a system reset. Although 8223 PROM s with pullup resistors are shown, a tri-state PROM such as an 82123 can be used without the resistors.

| TIME STATE | ROM ADDRESS | CPU-1 RAW CE | RAW ADDR ENAB | RAW MDR STROBE | NEXT ROM ADDR 16 | NEXT ROM ADDR 8 | NEST ROM ADDR 4 | NEXT ROM ADDR 2 | NEXT ROM ADDR 1 | CPU-2 PRIORITY STROBE | CPU DATA IN ENAB | CPU LOAD REG | CPU FLAG ADDR STB | RAW WRITE ENAB | RAW DATA IN ENAB | RAW DATA OUT STB | RAW DATA OUT ENAB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 14 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 2 | 30 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 3 | 26 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 4 | 27 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 5 | 19 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 6 | 23 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 7 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 8 | 15 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 9 | 11 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 10 | 9 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 11 | 25 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 12 | 29 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 13 | 21 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 14 | 5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 15 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 16 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 17 | 8 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 18 | 12 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 19 | 28 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 20 | 20 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 21 | 16 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 22 | 18 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 23 | 8 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |

**Figure 4. *Timing ROMS in Time Sequence***

System reset and power up control are handled by the two 7413 Schmidt triggers and other discrete circuitry at the bottom of figure 3. The first 7413 gives a snap-action response to BUS POWER OK which may be a slowly changing signal. The R-C network and second 7413 provide a signal that tracks BUS POWER OK but with a several millisecond delay. This delayed signal, after inversion, becomes BUS RESET. The transistors apply −12 volt power to the IMP chips when bus power is OK and remove it otherwise. BUS RESET also controls application of the 4-phase clocks to the microprocessor. Thus the timing relationship between power application and removal and clock application and removal is such that the IMP is properly initialized.

The logic in the upper third of figure 4 modifies some of the timing signals from the PROM according to bus cycle type: read or write. Flip-flop 1 samples BUS WRITE REQUEST at the leading edge of BUS CE and retains the read/write decision for the remainder of the cycle. The network at the top of the page consisting of a 7432 and 7410 delays the fall of BUS CE by 50ns during write cycles. It behaves as a simple inverter during read cycles. Lengthening BUS CE during write cycles only provides improved timing margins for writing into 4K RAM's without unnecessary power dissipation during read cycles. The gates on BDOS and BDOE gate these signals on for read cycles and off for write cycles. Likewise, BDIE and BWE are gated on for writes and off for reads.

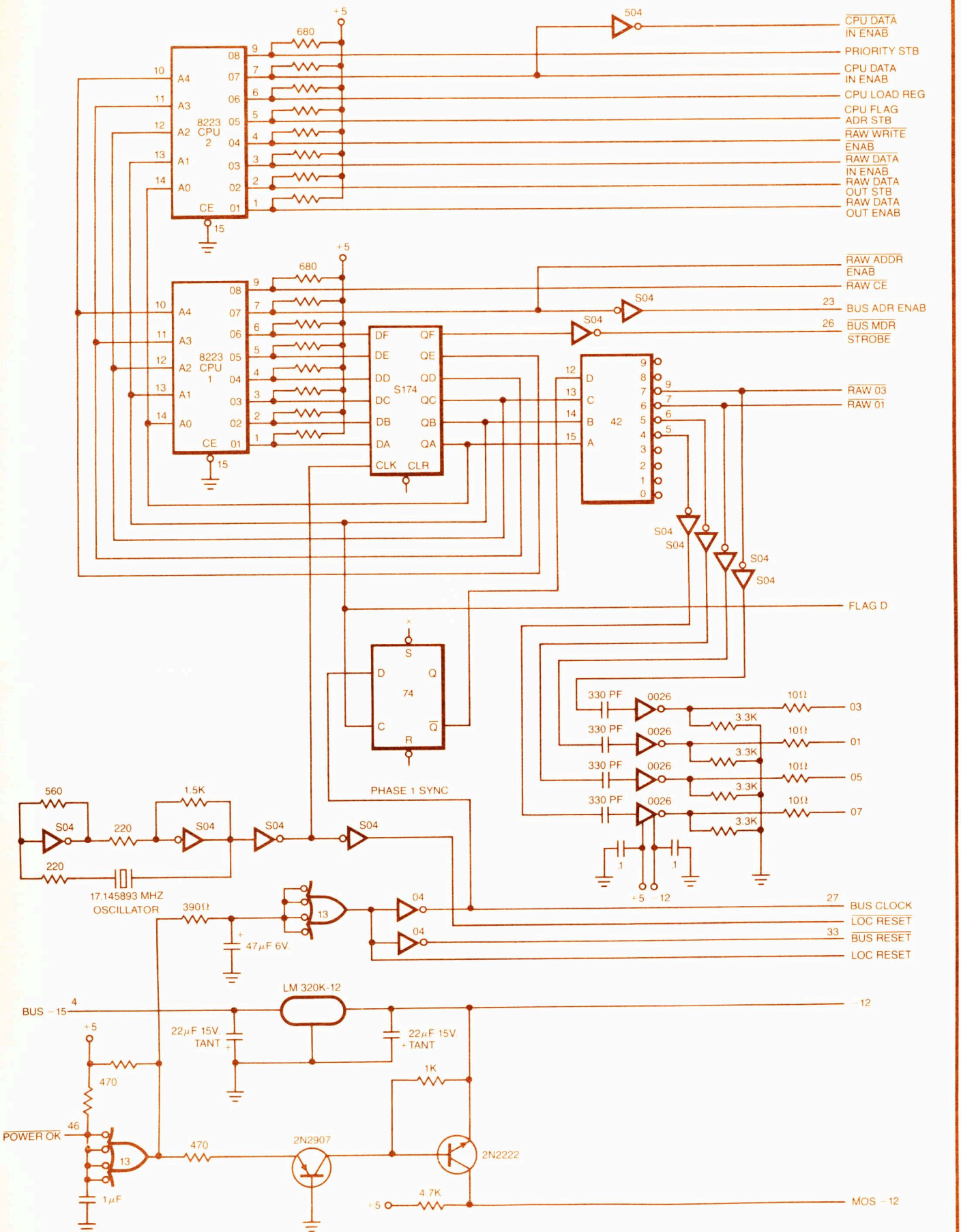



**Figure 5. *Timing Generator***

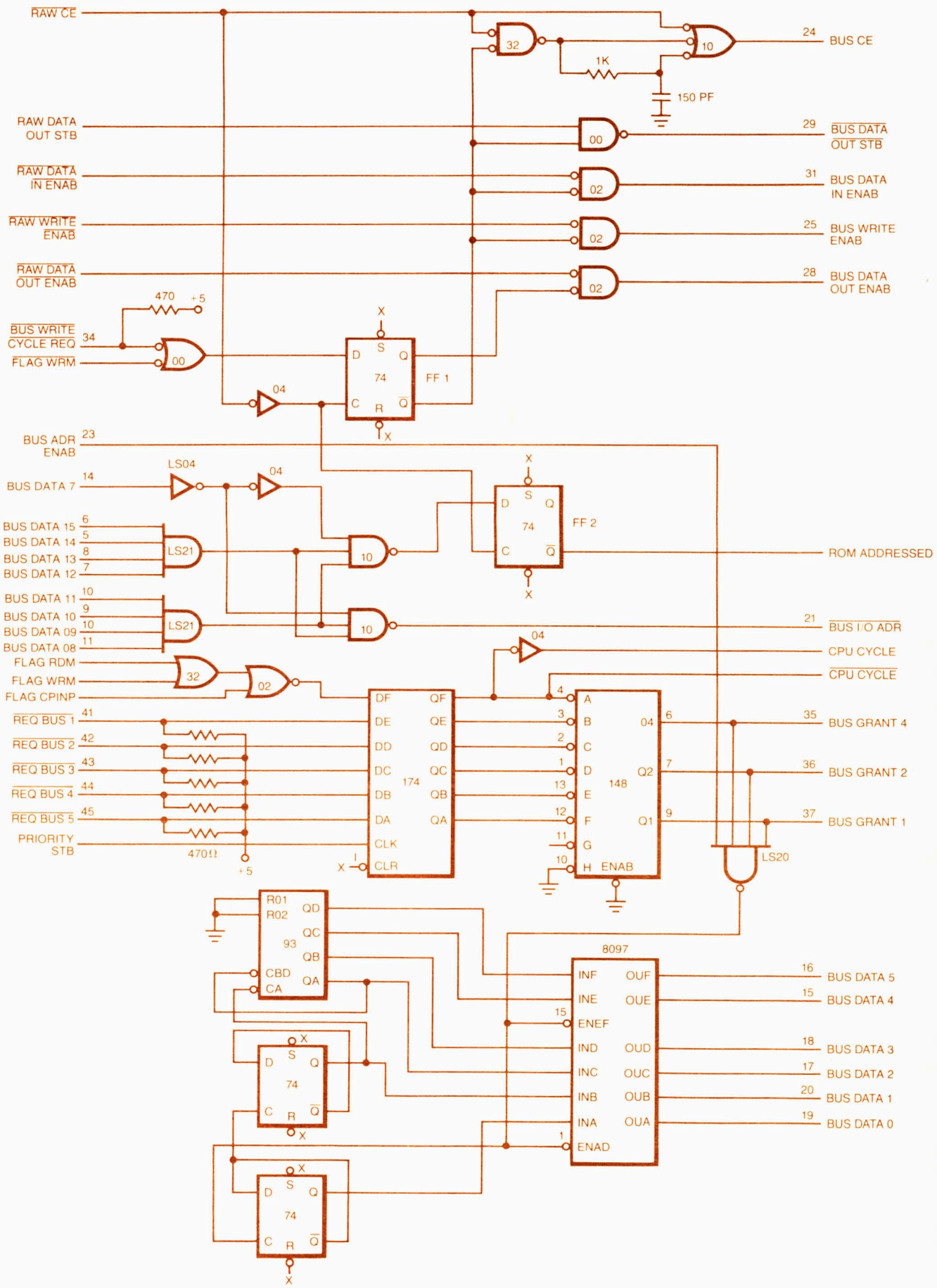



**Figure 6. *BUS Controller***

The network starting with the 74LS21's is a partial address decoder. If the address on the bus is between FF80 and FFFF, flip-flop 2 is set indicating that the on-board bootstrap ROM has been addressed. If the address is between FF00 and FF7F, $\overline{\text{BUS IIO ADDR}}$ is activated to inform peripherals that an I/O address is on the bus.

The next group of logic is the cycle request and grant priority logic. Gating for CPU cycle request and the 5 DMA request lines go into a hex latch that is strobed by PRIORITY STB near the end of each bus cycle. The latches are necessary to hold the input to the priority encoder constant throughout the next cycle. The 74148 determines the highest priority input present (active low, A is highest, H is lowest priority) and outputs a 3-bit code identifying that input. The G input is not used in this drawing but could be used for a sixth DMA request along with a latch. The H input is refresh request which is always present.

The bottom of figure 4 is the refresh logic for all dynamic memory in the system. A 74LS20 detects the coincidence of refresh grant (111) and BAE which indicates that the refresh address should be placed on the bus. The output thus enables an 8097 which gates the 6 significant refresh bits onto the bus. The other 10 bits assume a logic 1 and the bus controller assumes a read cycle. When the 8097 is gated off again, a 6 bit counter made from a 7474 and 7493 counts up one notch in preparation for the next refresh cycle. Two 8556 tri-state counters could have replaced the 7474, 7493, and 8097 used here but they were too hard to get to justify their use.

That concludes the description of the bus controller. Everything else in the system is just a collection of bus interfaces. Although the remainder of this series will be specifically concerned with IMP-16 interfacing to the bus, the basic concepts and bus structure can be used with any microprocessor. In fact, an essentially identical bus system was used in the design of a super 8008 system over three years ago.

In the next issue a brief description of the IMP-16 chip set will be given along with the remainder of the CPU board schematic and accompanying discussion.

# the Bit • Bucket

Dear Georgia:

In answer to your request for information on "cheap prom erasers", Byte Magazine, *May 1976* issue, published an article on a 1702A PROM programmer, in which the use of a GE # G8T5 ultra violet (germicidal) lamp was discussed. I have successfully used this lamp on the MM5203 chip. The chips were exposed for 6 minutes or longer at a distance of .25". Use caution when this lamp is on, protect your eyes and skin from exposure to the ultra violet rays.

Please add Systems & Services to your microcomputer consultant list. S&S is currently involved in PACE and SC/MP applications. The SC/MP kit has been buffered and is driving a front panel with address and instruction lamps. The front panel has run, step, stop and clear function. Future SC/MP expansions include 2K X 8 RAM and PROM cards, discrete line input/output cards and loaders which work with the "KITBUG" software.

Georgia, would you send me Vol. 2 No. 2 of the Bit • Bucket, it seems I missed that one.

Thanks.
Yours truly,

G. E. (Buz) Koenig
Systems & Services
P.O. Box 961
Hurst, Texas 76053

Dear Georgia:

Please send any information you have on your new high level language SM/PL. If no information is now available advise as to when the compiler will be released. We are almost ready to start programming our IMP-16P for an automatic diode testing application.

Thank you.
Very truly yours,
William R. Walters, C.E.E.
CODI Corporation
Pollitt Drive South
Fair Lawn, New Jersey 07410

*Ordering information for SM/PL will be announced this summer in the newsletter. SM/PL is a high-level language compiler for an IMP-16 with a minimum of 16K RAM.*

# EDITORIAL! EDITORIAL!

**by Dave Graves, Editor**

During the past months we have received several requests (at least one) for a classified section in COMPUTE. Georgia and I think it's a good idea. But there are some ground rules. First, the ads should pertain to microcomputers. We won't accept ads to sell your '64 Falcon. Second, the ads should be short. No novels allowed unless written by the editors. Third, if you are selling a product, we'll be glad to run your ad provided the product is useful to our members and it doesn't compete directly with National's products (sorry, but they pay the bills). And finally, since we are a little slow, if you submit dated material (schedules for classes, meetings, conferences, fairs, etc.) we should have it at least two months in advance of the event.

For all our patient readers who wait anxiously for each issue of COMPUTE, we are trying to get COMPUTE organized so you get it at the beginning of the month the issue is dated for. Won't that be a surprise!

COMPUTE/115
NATIONAL SEMICONDUCTOR CORP.
2900 SEMICONDUCTOR DRIVE
SANTA CLARA, CA. 95051